



## **HTTP Caching & Cache-Busting for Content Publishers**

**Michael J. Radwin**

<http://public.yahoo.com/~radwin/>

OSCON 2005  
Thursday, August 4th, 2005

## Agenda

- HTTP in 3 minutes
- Caching concepts
  - Hit, Miss, Revalidation
- 5 techniques for caching and cache-busting
- Not covered in this talk
  - Proxy deployment
  - HTTP acceleration (a.k.a. reverse proxies)
  - Database query results caching

2

YAHOO!

### Motivation:

Publishers have a lot of web content

HTML, images, Flash, movies

Speed is important part of user experience

Bandwidth is expensive

Use what you need, but avoid unnecessary extra

Personalization differentiates

Show timely data (stock quotes, news stories)

Get accurate advertising statistics

Protect sensitive info (e-mail, account balances)

### Not covered:

Proxy deployment is an interesting subject and deserves an entire lecture by itself

Configuring proxy cache servers (i.e. Squid)

Configuring browsers to use proxy caches

Transparent/interception proxy caching

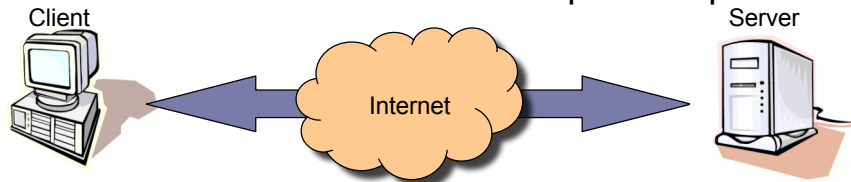
Inter-cache protocols (ICP, HTCP)

# HTTP and Proxy Review

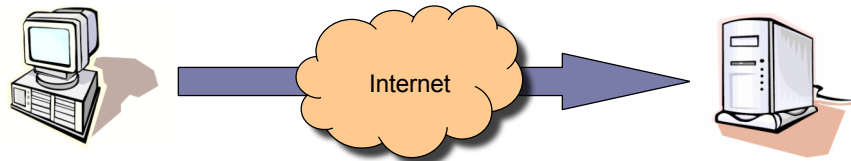


## HTTP: Simple and elegant

1. Client connects to `www.example.com` port 80

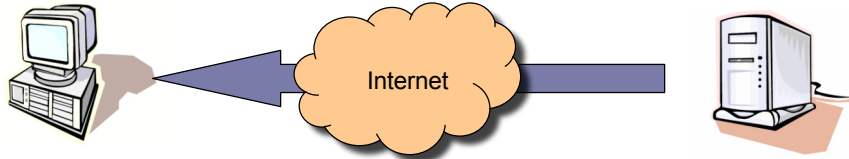


2. Client sends GET request

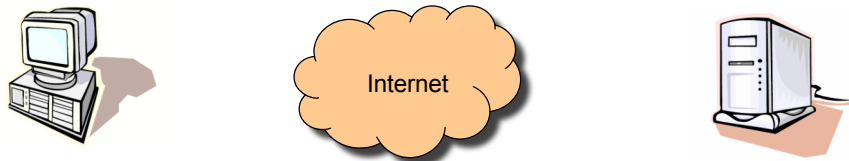


## HTTP: Simple and elegant

3. Server sends response



4. Client closes connection



## HTTP example

---

```
mradwin@machshav:~$ telnet www.example.com 80
```

```
Trying 192.168.37.203...
```

```
Connected to w6.example.com.
```

```
Escape character is '^]'.  
^C
```

```
GET /foo/index.html HTTP/1.1
```

```
Host: www.example.com
```

```
HTTP/1.1 200 OK
```

```
Date: Wed, 28 Jul 2004 23:36:12 GMT
```

```
Last-Modified: Thu, 12 May 2005 21:08:50 GMT
```

```
Content-Length: 3688
```

```
Connection: close
```

```
Content-Type: text/html
```

```
<html><head>
```

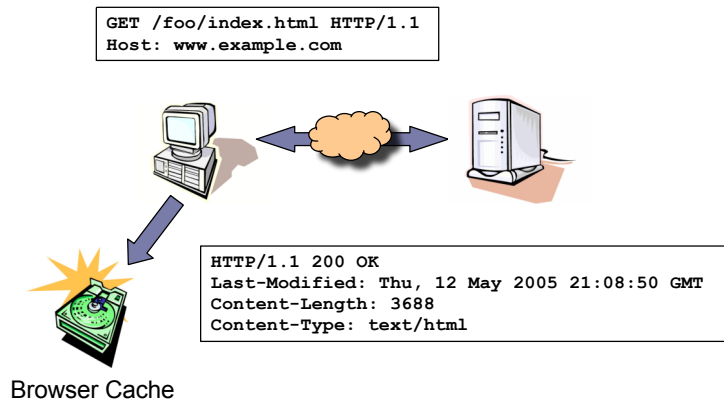
```
<title>Hello World</title>
```

```
...
```

6

YAHOO!

## Browsers use private caches



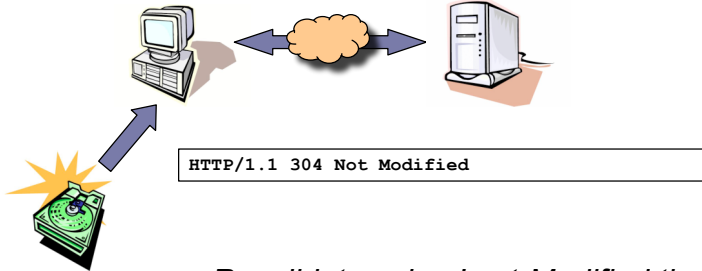
7

YAHOO!

Client stores copy of  
<http://www.example.com/foo/index.html> on its hard disk  
with timestamp.

## Revalidation (Conditional GET)

```
GET /foo/index.html HTTP/1.1
Host: www.example.com
If-Modified-Since: Thu, 12 May 2005 21:08:50 GMT
```



*Revalidate using Last-Modified time*

8

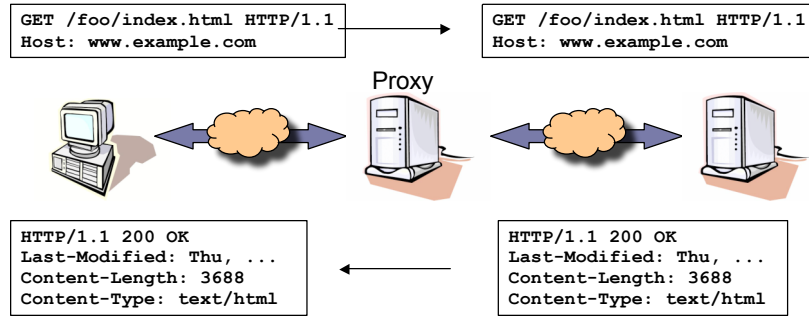
YAHOO!

The presence of If-Modified-Since header is what makes this a Conditional GET. Sometimes called an “IMS GET”.

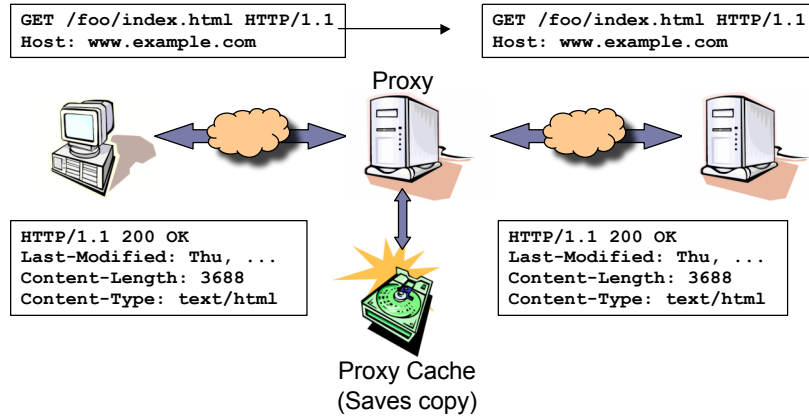
If content had actually changed, server would simply reply with a 200 OK and send full content.



# Non-Caching Proxy



## Caching Proxy: Miss



10

YAHOO!

## Caching Proxy: Hit

```
GET /foc/index.html HTTP/1.1  
Host: www.example.com
```



Proxy

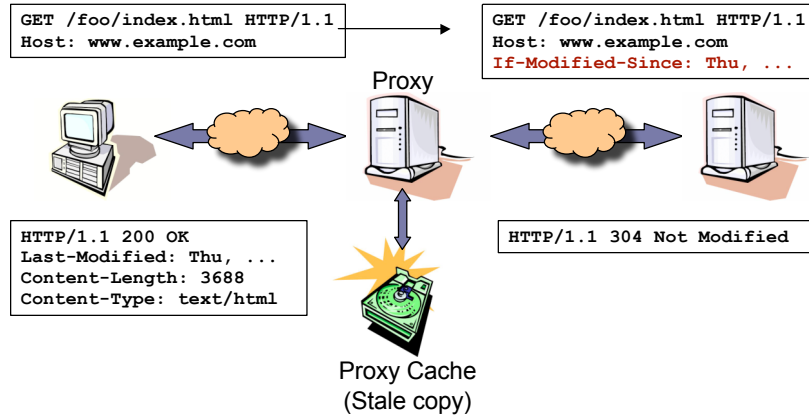


```
HTTP/1.1 200 OK  
Last-Modified: Thu, ...  
Content-Length: 3688  
Content-Type: text/html
```



Proxy Cache  
(Fresh copy!)

## Caching Proxy: Revalidation



12

YAHOO!



# Top 5 Caching Techniques

## Assumptions about content types

Rate of change once published		
<i>Frequently</i>	<i>Occasionally</i>	<i>Rarely/Never</i>
HTML	CSS JavaScript	Images Flash PDF
Dynamic Content Personalized		Static Content Same for everyone

## Top 5 techniques for publishers

---

1. Use **Cache-Control: private** for personalized content
2. Implement “Images Never Expire” policy
3. Use a cookie-free TLD for static content
4. Use Apache defaults for occasionally-changing static content
5. Use random tags in URL for accurate hit metering or very sensitive content

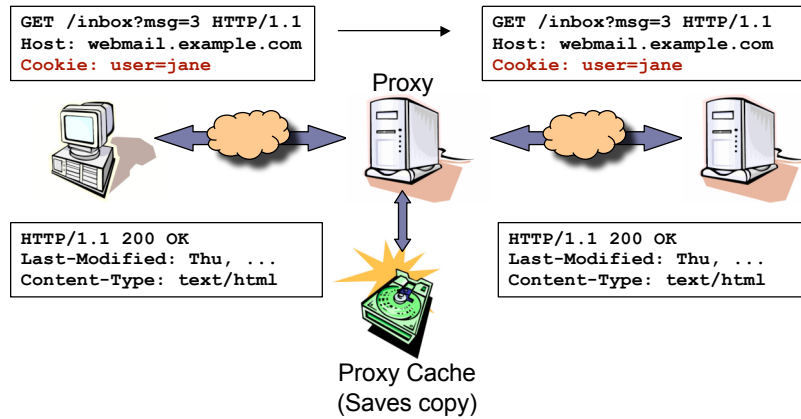
# 1. Cache-Control: private for personalized content

Rate of change once published		
<i>Frequently</i>	<i>Occasionally</i>	<i>Rarely/Never</i>
HTML	CSS JavaScript	Images Flash PDF
Dynamic Content Personalized		Static Content Same for everyone



## Bad Caching: Jane's 1st visit

- The URL isn't all that matters



## Bad Caching: Jane's 2nd visit

- Jane sees same message upon return

```
GET /inbox?msg=3 HTTP/1.1
Host: webmail.example.com
Cookie: user=jane
```



Proxy



```
HTTP/1.1 200 OK
Last-Modified: Thu, ...
Content-Type: text/html
```



Proxy Cache  
(Fresh copy of Jane's)

## Bad Caching: Mary's visit

- Witness a false positive cache hit

```
GET /inbox?msg=3 HTTP/1.1
Host: webmail.example.com
Cookie: user=mary
```



```
HTTP/1.1 200 OK
Last-Modified: Thu, ...
Content-Type: text/html
```



Proxy Cache  
(Fresh copy of Jane's)

## What's cacheable?

- HTTP/1.1 allows caching anything by default
  - Unless overridden with `Cache-Control` header
- In practice, most caches avoid anything with
  - `Cache-Control/Pragma` header
  - `Cookie/Set-Cookie` header
  - `WWW-Authenticate/Authorization` header
  - `POST/PUT` method
  - `302/307` status code (redirects)
  - SSL content

20

YAHOO!

### “13.4 Response Cacheability

Unless specifically constrained by a cache-control (section [14.9](#)) directive, a caching system MAY always store a successful response (see section [13.8](#)) as a cache entry, MAY return it without validation if it is fresh, and MAY return it after successful validation. If there is neither a cache validator nor an explicit expiration time associated with a response, we do not expect it to be cached, but certain caches MAY violate this expectation (for example, when little or no network connectivity is available). A client can usually detect that such a response was taken from a cache by comparing the Date header to the current time.”

## Cache-Control: private

- Shared caches bad for shared content
  - Mary shouldn't be able to read Jane's mail
- Private caches perfectly OK
  - Speed up web browsing experience
- Avoid personalization leakage with single line in httpd.conf or .htaccess

**Header set Cache-Control private**

21

YAHOO!

Note that HTTP/1.0 proxies aren't expected to understand Cache-Control header. If you're really concerned about user information leakage and there's a possibility that your users are behind HTTP/1.0 proxies, use technique #5 (random strings in the URL).

## 2. “Images Never Expire” policy

Rate of change once published		
<i>Frequently</i>	<i>Occasionally</i>	<i>Rarely/Never</i>
HTML	CSS JavaScript	Images Flash PDF
Dynamic Content Personalized		Static Content Same for everyone

## “Images Never Expire” Policy

- Dictate that images (icons, logos) once published never change
  - Set **Expires** header 10 years in the future
- Use new names for new versions
  - <http://us.yimg.com/i/new.gif>
  - <http://us.yimg.com/i/new2.gif>
- Tradeoffs
  - More difficult for designers
  - Faster user experience, bandwidth savings

23

YAHOO!

Pushing images to a separate server typically means that designers can't use 1-click publishing solutions such as Microsoft Frontpage.

## Imgs Never Expire: mod\_expires

```
# Works with both HTTP/1.0 and HTTP/1.1  
# (10*365*24*60*60) = 315360000 seconds
```

```
ExpiresActive On
```

```
ExpiresByType image/gif A315360000
```

```
ExpiresByType image/jpeg A315360000
```

```
ExpiresByType image/png A315360000
```

24

YAHOO!

$24 * 60 * 60 * 365 * 10 = 315360000$  seconds in ten years.

You may wish to add other mime types such as application/x-shockwave-flash



## Imgs Never Expire: mod\_headers

```
# Works with HTTP/1.1 only
<FilesMatch "\.(gif|jpe?g|png)$">
  Header set Cache-Control \
    "max-age=315360000"
</FilesMatch>

# Works with both HTTP/1.0 and HTTP/1.1
<FilesMatch "\.(gif|jpe?g|png)$">
  Header set Expires \
    "Mon, 28 Jul 2014 23:30:00 GMT"
</FilesMatch>
```

25

YAHOO!

You may wish to add other file extensions such as swf

Cache-Control is preferred for HTTP/1.1

Expires is for compatibility with HTTP/1.0 clients and proxies

When both headers are present, HTTP/1.1 clients typically prefer the Cache-Control header

## mod\_images\_never\_expire

```
/* Enforce policy with module that runs at URI translation hook */
static int translate_imgexpire(request_rec *r) {
    const char *ext;
    if ((ext = strrchr(r->uri, '.')) != NULL) {
        if (strcasecmp(ext, ".gif") == 0 || strcasecmp(ext, ".jpg") == 0 ||
            strcasecmp(ext, ".png") == 0 || strcasecmp(ext, ".jpeg") == 0) {
            if (ap_table_get(r->headers_in, "If-Modified-Since") != NULL ||
                ap_table_get(r->headers_in, "If-None-Match") != NULL) {
                /* Don't bother checking filesystem, just hand back a 304 */
                return HTTP_NOT_MODIFIED;
            }
        }
    }
    return DECLINED;
}
```

26

YAHOO!

Also <http://use.perl.org/~geoff/journal/22049>

### 3. Cookie-free static content

Rate of change once published		
<i>Frequently</i>	<i>Occasionally</i>	<i>Rarely/Never</i>
HTML	CSS JavaScript	Images Flash PDF
Dynamic Content Personalized		Static Content Same for everyone

## Use a cookie-free Top Level Domain for static content

- For maximum efficiency use 2 domains
  - `www.example.com` for dynamic HTML
  - `static.example.net` for images
- Many proxies won't cache `cookie` requests
  - But: multimedia is never personalized
  - Cookies irrelevant for images

28

YAHOO!

`static.example.com` won't cut it, because many cookies will be issued with “`domain=.example.com`”. Unless you're 100% sure you'll only issue cookies with “`domain=www.example.com`”, you'll need to use a completely different TLD. Yahoo!, for example, uses `yahoo.com` for dynamic HTML content and `yimg.com` for images and other static content.

## Typical GET request w/Cookies

```
GET /i/foo/bar/quux.gif HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.0; en-US; rv:1.7) Gecko/20040707
Firefox/0.8
Accept: application/x-shockwave-
flash,text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.
8,video/x-mng,image/png,image/jpeg,image/gif;q=0.2,*/*;q=0.1
Cookie: U=mt=vtC1tp2MhYv9RL5B1pxYRFN_P8DpMJoaMl1EcA--&ux=Iir.AB&un=42vnticvufc8v;
brandflash=1; B=amfco1503sqp8&b=2; F=a=NC184LcsvfX96G.JR27qSjCHu7bII3s.
tXa44psMLiFtVoJB_m5wecWY_.7&b=K1It; LYC=l_v=2&l_lv=7&l_l=h03m8d50c8bo
&l_s=3yu2qzx5zvqwuwuzv22wrwr5t3wlzsr&l_lid=14rsb76&l_r=a8&l_um=1_0_1_0_0;
GTSessionID835990899023=83599089902340645635; Y=v=1&n=6eecegejj7012f
&l=h03m8d50c8bo/o&p=m012o33013000007&jb=16|47|&r=a8&lg=us&intl=us&np=1;
PROMO=SOURCE=fp5; YGCV=d=; T=z=iTu.ABiZD/AB6dPwoqXibIcTzc0BjY3TzI3NTY0MzQ-
&a=YAE&sk=DAAwRz5H1DUN2T&d=c2wBT0RBekFURXdPRFV3TWpFek5ETS0BYQFZQUUBb2sBWLcwLQF0aXABW
UhaTVBBAXp6AW1UdS5BQmdXQQ--&af=QFQBQ0FDQURCOUF1QUJBQ0FEQUtBYTE
FNSDAmdHM9MTA5MDE4NDQxOCZwcz11OG83MUVYcTYxOVouT2Ftc1ZFZUhbLS0-;
LYS=l_fh=0&l_vo=myla; PA=p0=dg13DX4NdGk-&p1=6L5qmg--&e=xMv.AB;
YP.us=v=2&m=addr&d=1525+S+Robertson+Bld%01Los+Angeles%01CA%0190035-
4231%014480%0134.051590%01-118.384342%019%01a%0190035
Referer: http://www.example.com/foo/bar.php?abc=123&def=456
Accept-Language: en-us,en;q=0.7,he;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
```

29

YAHOO!

Since a Cookie header is sent, some proxies will refuse to cache the response.

## Same request, no Cookies

```
GET /i/foo/bar/quux.gif HTTP/1.1
Host: static.example.net
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.0; en-US; rv:1.7) Gecko/20040707
Firefox/0.8
Accept: application/x-shockwave-
      flash,text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.
      8,video/x-mng,image/png,image/jpeg,image/gif;q=0.2,*/*;q=0.1
Referer: http://www.example.com/foo/bar.php?abc=123&def=456
Accept-Language: en-us,en;q=0.7,he;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
```

- Bonus: much smaller GET request
  - Dial-up MTU size 576 bytes, PPPoE 1492
  - 1450 bytes reduced to 550

## 4. Apache defaults for static, occasionally-changing content

<i>Frequently</i>	Rate of change once published <i>Occasionally</i>	<i>Rarely/Never</i>
HTML	CSS JavaScript	Images Flash PDF
Dynamic Content Personalized		Static Content Same for everyone

## Revalidation works well

- Apache handles revalidation for static content
  - Browser sends `If-Modified-Since` request
  - Server replies with short `304 Not Modified`
  - No special configuration needed
- Use if you can't predict when content will change
  - Page designers can change immediately
  - No renaming necessary
- Cost: extra HTTP transaction for `304`
  - Smaller with `Keep-Alive`, but large sites disable

32

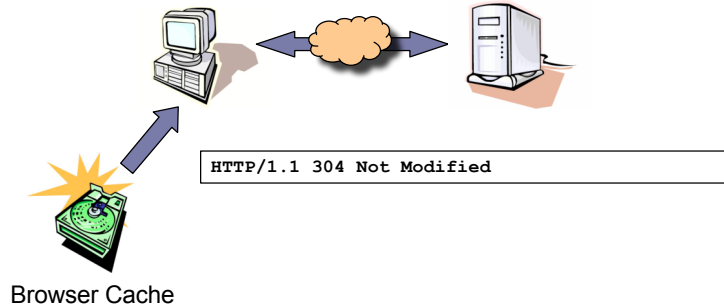
YAHOO!

Each HTTP request has some latency. When you disable Keep-Alive (as any large site typically must do to scale), each HTTP request requires a full 3-way TCP handshake. The handshake latency can be perceptible, especially on a slow connection such as a 56k modem.



## Successful revalidation

```
GET /foo/index.html HTTP/1.1
Host: www.example.com
If-Modified-Since: Thu, 12 May 2005 21:08:50 GMT
```



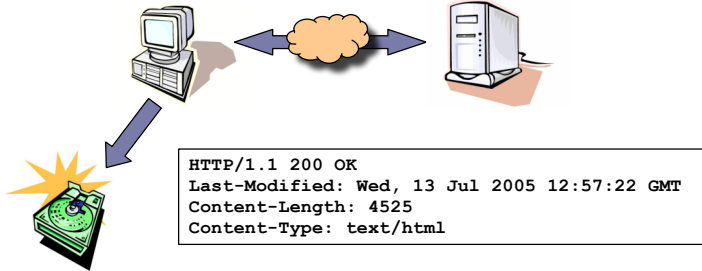
33

YAHOO!

Apache simply `stat()`s the file and compares the timestamp to the `If-Modified-Since` timestamp. If the file's timestamp is less than or equal to the `If-Modified-Since` header, it returns `304 Not Modified`.

## Updated content

```
GET /foo/index.html HTTP/1.1  
Host: www.example.com  
If-Modified-Since: Thu, 12 May 2005 21:08:50 GMT
```



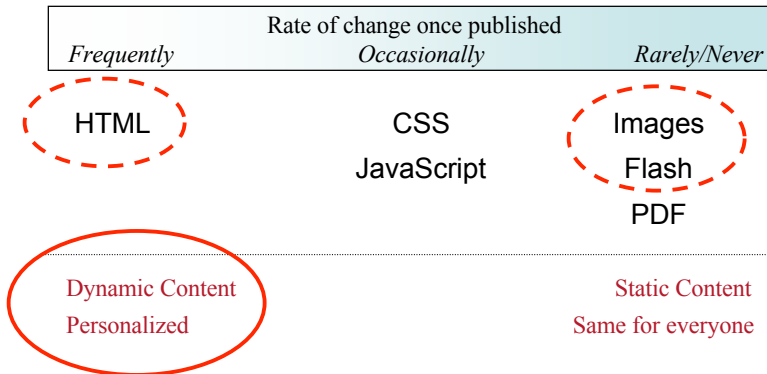
Browser Cache

34

YAHOO!

Content has been modified. Client tries to revalidate again, but revalidation fails because URI has been updated. Apache returns 200 OK with full content.

## 5. URL Tags for sensitive content, hit metering



## URL Tag technique

---

- Idea
  - Convert public shared proxy caches into private caches
  - Without breaking real private caches
- Implementation: pretty simple
  - Assign a per-user URL tag
  - No two users use same tag
  - Users never see each other's content

## URL Tag example

---

- Goal: accurate advertising statistics
- Do you trust proxies?
  - Send `Cache-Control: must-revalidate`
  - Count `304 Not Modified` log entries as hits
- If you don't trust 'em
  - Ask client to fetch tagged image URL
  - Return `302` to highly cacheable image file
  - Count `302s` as hits
  - Don't bother to look at cacheable server log

## Hit-metering for ads (1)

```
<script type="text/javascript">
var r = Math.random();
var t = new Date();
document.write("<img width='109' height='52'
  src='http://ads.example.com/ad/foo/bar.gif?t="
  + t.getTime() + ";r=" + r + "'>");
</script>
<noscript>
<img width="109" height="52" src=
  "http://ads.example.com/ad/foo/bar.gif?js=0">
</noscript>
```

38

YAHOO!

No, this is not RFC 2227, which uses headers like “Connection: meter” and “Meter: count=1/0”

## Hit-metering for ads (2)

```
GET /ad/foo/bar.gif?t=1090538707;r=0.510772917234983 HTTP/1.1
Host: ads.example.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.0; en-US; rv:1.7)
           Gecko/20040707 Firefox/0.8
Referer: http://www.example.com/foo/bar.php?abc=123&def=456
Cookie: uid=C50DF33E-E202-4206-B1F3-946AEDF9308B
```

```
HTTP/1.1 302 Moved Temporarily
Date: Wed, 28 Jul 2004 23:45:06 GMT
Location: http://static.example.net/i/foo/bar.gif
Content-Type: text/html
```

```
<a href="http://static.example.net/i/foo/bar.gif">Moved</a>
```

## Hit-metering for ads (3)

```
GET /i/foo/bar.gif HTTP/1.1
Host: static.example.net
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.0; en-US; rv:1.7)
Gecko/20040707 Firefox/0.8
Referer: http://www.example.com/foo/bar.php?abc=123&def=456
```

```
HTTP/1.1 200 OK
Date: Wed, 28 Jul 2004 23:45:07 GMT
Last-Modified: Mon, 05 Oct 1998 18:32:51 GMT
ETag: "69079e-ad91-40212cc8"
Cache-Control: public,max-age=315360000
Expires: Mon, 28 Jul 2014 23:45:07 GMT
Content-Length: 6096
Content-Type: image/gif
```

```
GIF89a...
```



## URL Tags & user experience

---

- Does not require modifying HTTP headers
  - No need for `Pragma: no-cache` Or `Expires` in past
  - Doesn't break the Back button
- Browser history & visited-link highlighting
  - JavaScript timestamps/random numbers
    - Easy to implement
    - Breaks visited link highlighting
  - Session or Persistent ID preserves history
    - A little harder to implement

## Breaking the Back button

---

- User expectation: Back button works instantly
  - Private caches normally enable this behavior
- Aggressive cache-busting breaks Back button
  - Server sends `Pragma: no-cache` Or `Expires` in past
  - Browser must re-visit server to re-fetch page
  - Hitting network much slower than hitting disk
  - User perceives lag
- Use aggressive approach very sparingly
  - Compromising user experience is A Bad Thing



## Review: Top 5 techniques

---

1. Use **Cache-Control: private** for personalized content
2. Implement “Images Never Expire” policy
3. Use a cookie-free TLD for static content
4. Use Apache defaults for occasionally-changing static content
5. Use random tags in URL for accurate hit metering or very sensitive content

## Pro-caching techniques

- **Cache-Control: max-age=<bignum>**
- **Expires:** <10 years into future>
- Generate “static content” headers
  - **Last-Modified, ETag**
  - **Content-Length**
- Avoid “cgi-bin”, “.cgi” or “?” in URLs
  - Some proxies (e.g. Squid) won’t cache
  - Workaround: use **PATH\_INFO** instead

45

YAHOO!

In other words, these are ways to make dynamic content look like static content.

## Cache-busting techniques

- Use POST instead of GET
- Use random strings and “?” char in URL
- Omit `Content-Length` & `Last-Modified`
- Send explicit headers on response
  - Breaks the back button
  - Only as a last resort

`Cache-Control: max-age=0,no-cache,no-store`

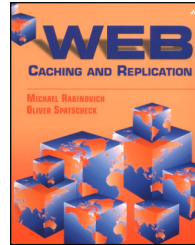
`Expires: Tue, 11 Oct 1977 12:34:56 GMT`

`Pragma: no-cache`

## Recommended Reading

- **Web Caching and Replication**

- Michael Rabinovich & Oliver Spatscheck
- Addison-Wesley, 2001



- **Web Caching**

- Duane Wessels
- O'Reilly, 2001



Slides: <http://public.yahoo.com/~radwin/>



**LIFE ENGINE™**