UNIVERSITY OF
Southampton

UNIVERSITY OF Southampton

Content Negotiation, Conditional Requests and Cookies

COMP3220 Web Infrastructure

Dr Nicholas Gibbins – nmg@ecs.soton.ac.uk

UNIVERSITY OF Southampton

Content Negotiation

HTTP content negotiation

HTTP can deliver different representations of a resource based on client preferences

Two styles of content negotiation:

- Server-driven (where the server makes the final choice of representation)
- Client-driven (where the client makes the final choice of representation)

Three areas for negotiation

- Media type (`Accept:` and `Content-Type:`)
- Language (`Accept-Language:` and `Content-Language:`)
- Encoding (`Accept-Encoding:` and `Content-Encoding:`)

Server-driven content negotiation

1. Client tells the server what it is able to accept in a request header
2. Server chooses an appropriate representation to return to the client
3. Server tells the client what its choice was in a response header

Client request headers all have the same structure:

headername: $choice_1$; q=$quality_1$, $choice_2$; q=$quality_2$, $choice_3$; q=$quality_3$ ...

The header name and the values for $choice_n$ all depend on what is being negotiated

The quality values $quality_n$ are numeric values between 0.0 and 1.0

Media type negotiation

Client request header: `Accept:`

Server response header: `Content-Type:`

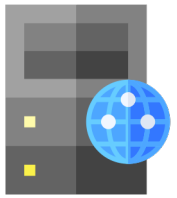
Choice values are Internet Media Types:

- `text/plain`
- `text/html`
- `application/pdf`
- `image/jpeg`
- `image/png`
- `image/svg+xml`
- `audio/mpeg`
- …

Media type negotiation

```
GET / HTTP/1.1
Host: example.org
Accept: text/html; q=1.0, text/plain; q=0.5
```

```
HTTP/1.1 200 OK
Content-Type: text/html

<html>
  <head>
    <title>Example, Inc. Homepage</title>
  </head>
  <body><h1>Welcome to Example!</h1>...</body>
</html>
```



Language negotiation

Client request header: `Accept-Language:`

Server response header: `Content-Language:`

Choice values are typically ISO 639 language codes:

- `en` for English
- `de` for German
- `zh` for Chinese

Convention for indicating regional dialects by adding ISO 3166-1 country codes:

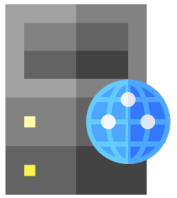
- `en-GB` for British English
- `de-CH` for Swiss German

Phillips, A. and Davis, M. (2009) *Tags for identifying languages*. RFC5646/BCP47. Available at: https://tools.ietf.org/html/bcp47

Language negotiation

```
GET / HTTP/1.1
Host: example.org
Accept-Language: de; q=1.0, en-gb; q=0.5
```

```
HTTP/1.1 200 OK
Content-Type: text/html
Content-Language: de

<html>
  <head>
    <title>Example, Inc. Homepage</title>
  </head>
  <body><h1>Willkommen zu Example!</h1></body>
</html>
```



Encoding negotiation

Client request header: `Accept-Encoding:`

Server response header: `Content-Encoding:`

Typically used to specify whether the body of a HTTP message is compressed

Possible choice values:

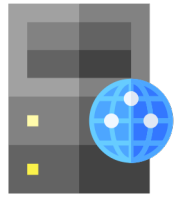
- `br` (Brotli compressed data format)
- `deflate` (zlib compression)
- `gzip` (LZ77 compression – the most common choice)
- `compress` (LZW compression)
- `identity` (no encoding)

The `Vary:` header

Origin servers can indicate what parts of the request message (aside from the method, `Host:` header, and request target) they'll use to select an appropriate representation

```
GET / HTTP/1.1
Host: example.org
```

```
HTTP/1.1 200 OK
Content-Type: text/html
Vary: accept, accept-language

...
```

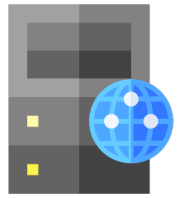


The `Content-Location:` header

Used to indicate a direct URI to use to access a content-negotiated resource

```
GET /logo HTTP/1.1
Host: example.org
Accept: image/png; q=1.0, image/gif; q=1.0, image/jpeg; q=0.8
```

```
HTTP/1.1 200 OK
Content-Type: image/gif
Content-Location: /logo.gif

...
```



Client-driven content negotiation

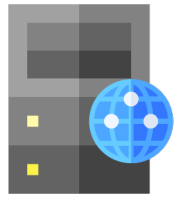
1. Client requests a resource representation
2. Server returns 300 `Multiple Choices` with a list of URIs for alternative representations in the body
3. Client requests a representation of one of those URIs

No standard for how the server lists the alternative representations!

Client-driven content negotiation

```
GET / HTTP/1.1
Host: example.org
```

```
HTTP/1.1 300 Multiple Choices
Content-Type: application/json

[{"type": "image/jpeg",
  "uri": "http://example.org/me.jpg"},
 {"type": "image/png",
  "uri": "http://example.org/me.png"}]
```

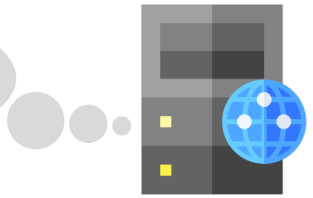


```
GET / HTTP/1.1
Host: example.org
```

How not to negotiate content

```
GET / HTTP/1.1
Host: example.org
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15;
rv:80.0) Gecko/20100101 Firefox/80.0
```

You're a Firefox browser, so I'd better send you the Firefox version of that resource



How not to negotiate content (Browser Wars, Part II)

Client Hints

Content negotiation using the `User-Agent:` header is not considered good practice

`Accept:` is a very crude basis for content negotiation

Client Hints is a proposed HTTP extension that allows browsers to state their capabilities and preferences

- Device memory
- Downlink speed
- Viewport width
- Device pixel ratio
- Reduced data usage

Grigorik, I. and Weiss, Y. (2020) *HTTP Client Hints*. Internet Draft. Available online at: https://httpwg.org/http-extensions/client-hints.html

Exercise: Content Negotiation

Exercise: Content Negotiation

`curl -H "[header]" [uri]`

- Generates a HTTP request including the specified header

Use curl to study the following requests:

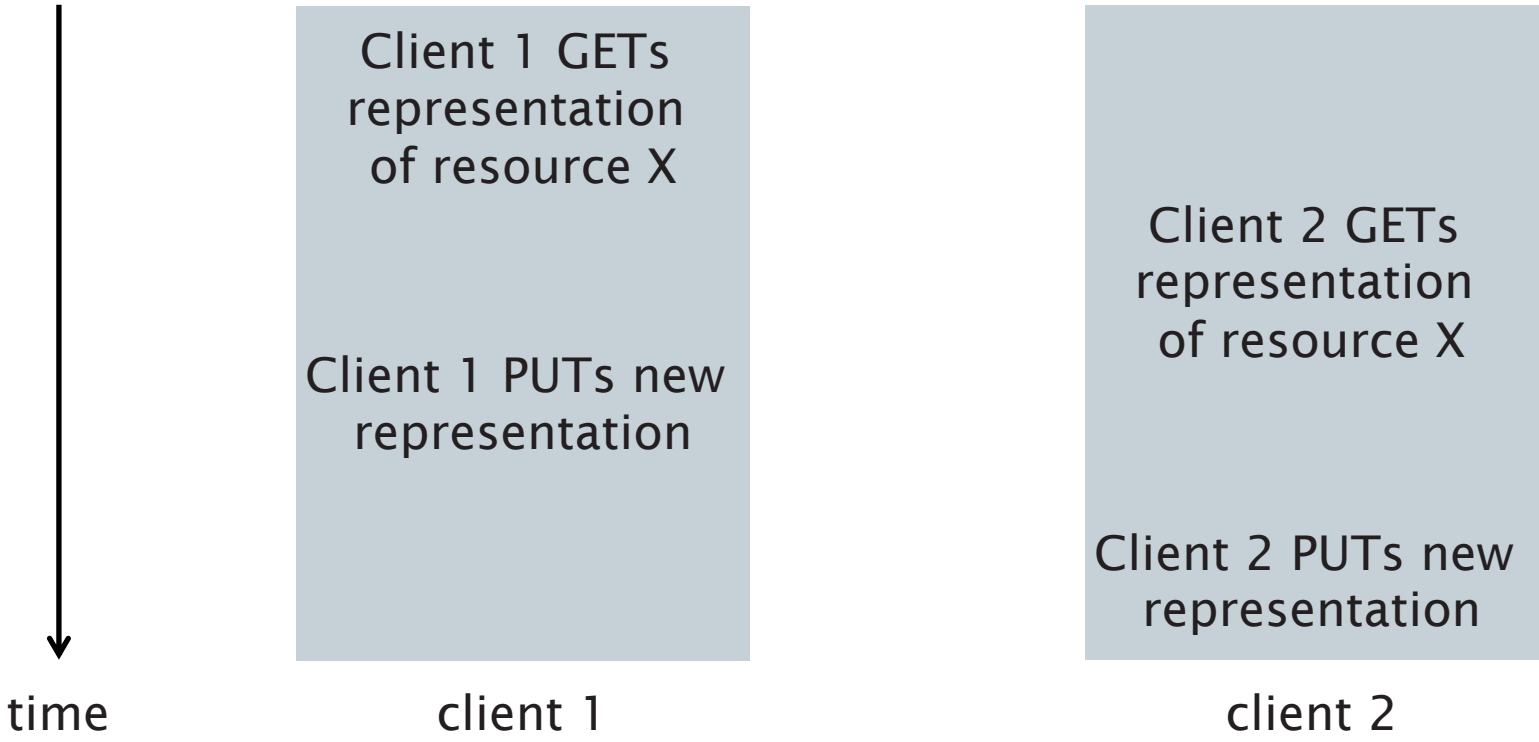
```
curl -v https://www.debian.org/
curl -v -H "Accept-Language: de" https://www.debian.org/
curl -v -H "Accept-Language: fi" https://www.debian.org/
```

UNIVERSITY OF Southampton

Conditional Requests

The Lost Update problem

If two or more clients are modifying a resource at the same time, how do we avoid *lost updates*?



Conditional requests

Some HTTP methods are *unsafe*; they alter the state of resources

Lost updates occur only when carrying out unsafe methods

- Avoid by checking to see whether the state of a resource (i.e. its representation) has changed between the GET and the PUT

Can also be used to avoid unnecessary safe methods

Comparison of representations in terms of validators that describe resource versions

- Strong validation: checks whether representations are byte-for-byte identical
- Weak validation: checks whether representations contain “the same content”

Entity tags

Entity tags are identifiers for resource versions (can be weak or strong)

Supplied by the server using the ETag: header:

- `ETag: "38be-5af769c088685"` - a strong entity tag
- `ETag: W/"0023"` - a weak entity tag (denoted by the “W/”)
- `ETag: *` - an entity tag matching any resource

`If-Match: <etag>, <etag>, ...`

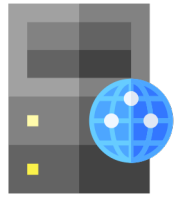
- Only carry out the operation if the resource entity tag matches one of those listed
- If no matches, return `412 Precondition Failed`

`If-None-Match: <etag>, <etag>, ...`

- Only carry out the operation if the resource entity tag does not match any of those listed
- If matches, return `304 Not Modified` for safe methods, or `412 Precondition Failed` for unsafe methods

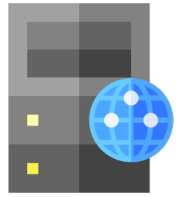
```
GET / HTTP/1.1
Host: www.example.com
```

```
HTTP/1.1 200 OK
Content-Type: text/html
ETag: "39d5-5943f8fdc2607"

<representation>
```



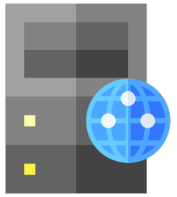
```
PUT / HTTP/1.1
Host: www.example.com
If-Match: "39d5-5943f8fdc2607"

<new representation>
```

```
HTTP/1.1 200 OK
Content-Type: text/html
ETag: "47cf-b922e4f191138"
```



```
GET / HTTP/1.1
Host: www.example.com
```

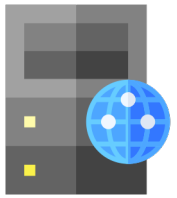
```
HTTP/1.1 200 OK
Content-Type: text/html
ETag: "39d5-5943f8fdc2607"

<representation>
```



http://www.example.com/ is modified – new ETag is 4b4f-77a6c3ab117a2

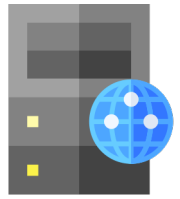
```
PUT / HTTP/1.1
Host: www.example.com
If-Match: "39d5-5943f8fdc2607"

<new representation>
```

```
HTTP/1.1 412 Precondition Failed
Content-Type: text/html
ETag: "4b4f-77a6c3ab117a2"
```



```
GET / HTTP/1.1
Host: www.example.com
If-None-Match: "4b4f-77a6c3ab117a2"
```

```
HTTP/1.1 304 Not Modified
```



Timestamps

Supplied by the server using the `Last-Modified:` header:

- `Last-Modified: Fri, 18 Sep 2020 17:25:23 GMT`

`If-Modified-Since: <dayname>, <day> <month> <year> <hour>:<min>:<sec> GMT`

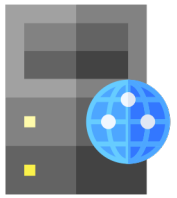
- Only carry out the operation if the resource timestamp is after the given date
- Return `304 Not Modified` if before the given date

`If-Unmodified-Since: <dayname>, <day> <month> <year> <hour>:<min>:<sec> GMT`

- Only carry out the operation if the resource timestamp is not after the given date
- Return `412 Precondition Failed` if after the given date

```
GET / HTTP/1.1
Host: www.example.com
```

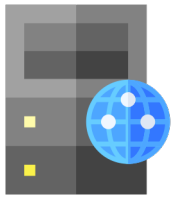
```
HTTP/1.1 200 OK
Content-Type: text/html
Last-Modified: Mon, 14 Sep 2020 12:23:01 GMT

<representation>
```



http://www.example.com/ is modified on Tue, 15 Sep 2020 14:34:07 GMT

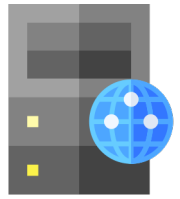
```
PUT / HTTP/1.1
Host: www.example.com
If-Unmodified-Since: Mon, 14 Sep 2020 12:23:01 GMT

<new representation>
```

```
HTTP/1.1 412 Precondition Failed
Content-Type: text/html
Last-Modified: Tue, 15 Sep 2020 14:34:07 GMT
```



```
GET / HTTP/1.1
Host: www.example.com
If-Modified-Since: Tue, 15 Sep 2020 14:34:07 GMT
```

```
HTTP/1.1 304 Not Modified
```



Further reading

Fielding, R. and Reschke, J. (2014) *Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests*. RFC7232.

https://tools.ietf.org/html/rfc7232

UNIVERSITY OF Southampton

Cookies

The infamous cookie

Intended as a way for web servers to persist state across HTTP requests

- (but isn't HTTP supposed to be stateless?)

Invented in 1994 by Lou Montulli of Netscape

- (also the inventor of the <blink> tag)

Patented in 1995

- (Netscape Communications Corp. vs ValueClick Inc.)

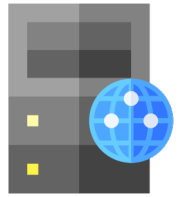
Ostensibly deprecated in favour of HTML5 local storage ... and yet it still lives!

Barth, A. (2011) *HTTP State Management Mechanism*. RFC6265. Available online at: https://tools.ietf.org/html/rfc6265

Cookies

```
GET / HTTP/1.1
Host: www.example.org
```

```
HTTP/1.1 200 OK
Content-Type: text/html
Set-Cookie: foo=23
Set-Cookie: bar=qux

...
```



```
GET / HTTP/1.1
Host: www.example.org
Cookie: foo=23; bar=qux
```

Lifetime

Cookies can either be session cookies or permanent cookies

- Session cookies expire "when the current session ends" (when the browser is closed?)
- Permanent cookies have a specified expiry time

```
Set-Cookie: foo=23; Expires=Fri, 23 Oct 2020 10:05:34 GMT
```

- Indicates that the cookie should expire after the given date

```
Set-Cookie: foo=23; Max-Age=21600
```

- Indicates that the cookie should expire after `Max-Age` seconds

Access restriction

`Set-Cookie: foo=23; Secure`

- Indicates that the cookie should only ever be sent over HTTPS

`Set-Cookie: foo=23; HttpOnly`

- Indicates that the cookie should not be visible from within the `Document.cookie` interface

Scope

`Set-Cookie: foo=23; Domain=example.org`

- Cookie should only be sent to example.org or its subdomains (i.e. foo.example.org)
- Defaults to the origin server that set the cookie (excluding subdomains)

`Set-Cookie: foo=23; Path=/bar`

- Cookie should only be sent if value of `Path` is in the requested URI

`Set-Cookie: foo=23; SameSite=Strict`

- `Strict`: only send cookie to the same site that originated it
- `Lax`: cookie is withheld for cross-site subrequests (i.e. images) but sent when user follows a link (typical browser default setting)
- `None`: no restrictions on cross-site requests

Privacy considerations

Cookies can be used to track users across websites

Relies on websites embedding resources (typically images) from third parties

- When a user fetches the third party resource, it sets a cookie
- Cookie may be read when the user fetches a resource from the third party in future

Relies on default browser cookie scope of `SameSite=None`

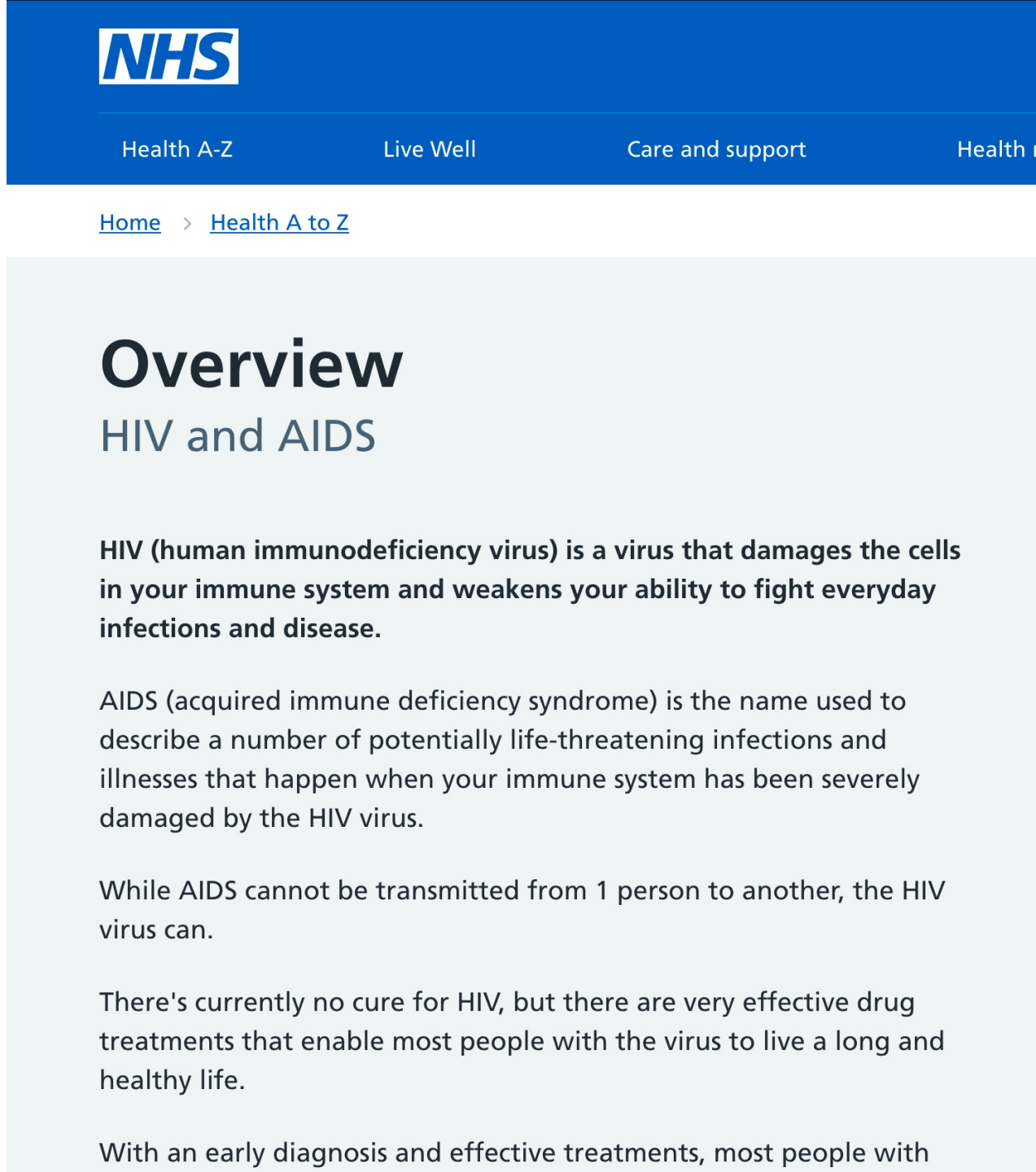

GET /page.html
200 OK
site1.org
page.html
foo.gif
GET /foo.gif
200 OK
Set-Cookie: id=12345
bigbrother.com
foo.gif
GET /doc.html
200 OK
site2.org
doc.html
bar.gif
GET /bar.gif
Cookie: id=12345
200 OK
bigbrother.com
bar.gif

Privacy considerations

High profile UK case in 2010
(discovered by a Southampton graduate!)

- NHS put Facebook "like" buttons on their webpages (loaded from Facebook's CDN)
- When a user visited an NHS webpage, Facebook set a cookie
- If the user visited a different page with a like button, Facebook could read the cookie and correlate those visits
- If they were logged into Facebook, Facebook now knew what their users are searching the NHS for

https://mmt.me.uk/blog/2010/11/21/nhs-and-tracking/



UNIVERSITY OF Southampton

Next Lecture: HTML