Electronics and
Computer Science

UNIVERSITY OF Southampton

Web Protocols: HTTP

COMP6017 Topics on Web Services

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

- 2013-2014

Web Protocols

Many protocols in use on the Web, but only two are Web protocols

- Hypertext Transfer Protocol
- Simple Object Access Protocol

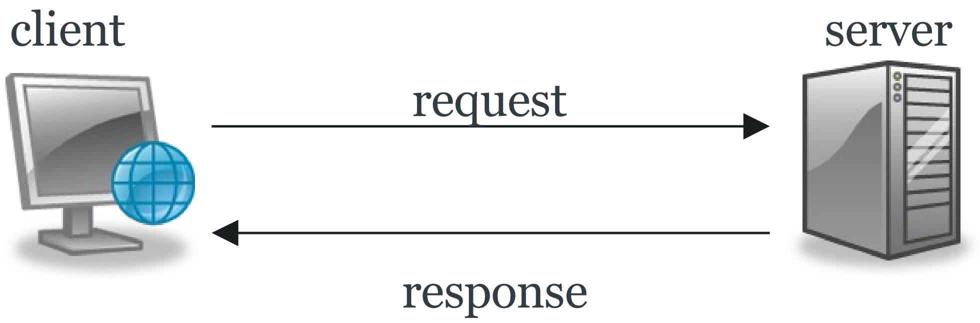
HTTP: Hypertext Transfer Protocol

Hypertext Transfer Protocol

Application protocol for distributed hypermedia

- First documented in 1991 (HTTP/0.9)
- HTTP/1.0 introduced in 1996 (RFC1945)
- HTTP/1.1 last updated in 1999 (RFC2616)

Client and server exchange request/response messages



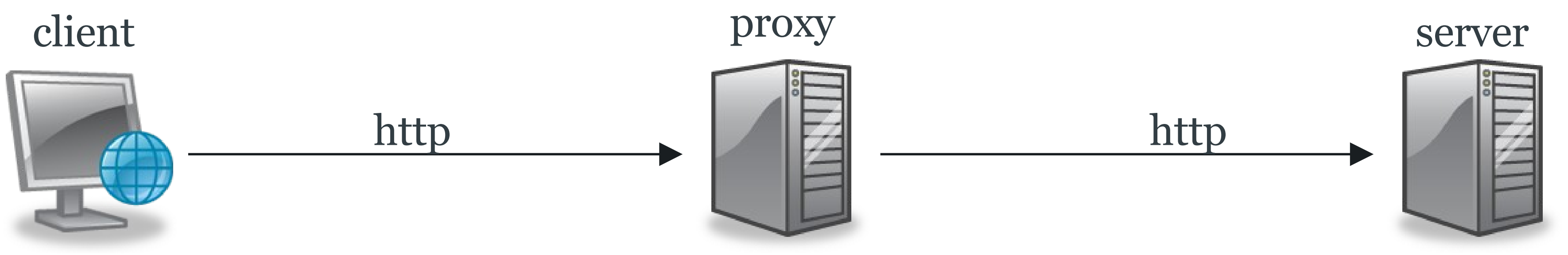
Hypertext Transfer Protocol

Typically a direct connection between client and server

May be intermediaries in the request/response chain

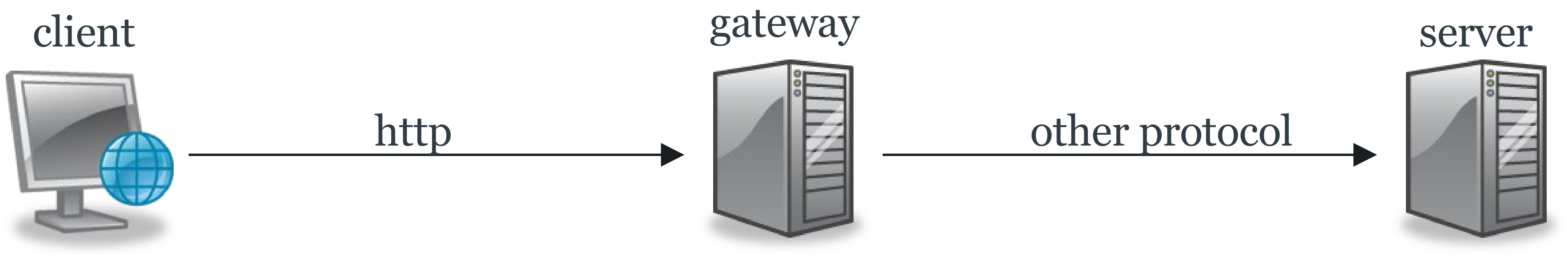
- Proxy
- Gateway
- Tunnel

HTTP Intermediaries: Proxy



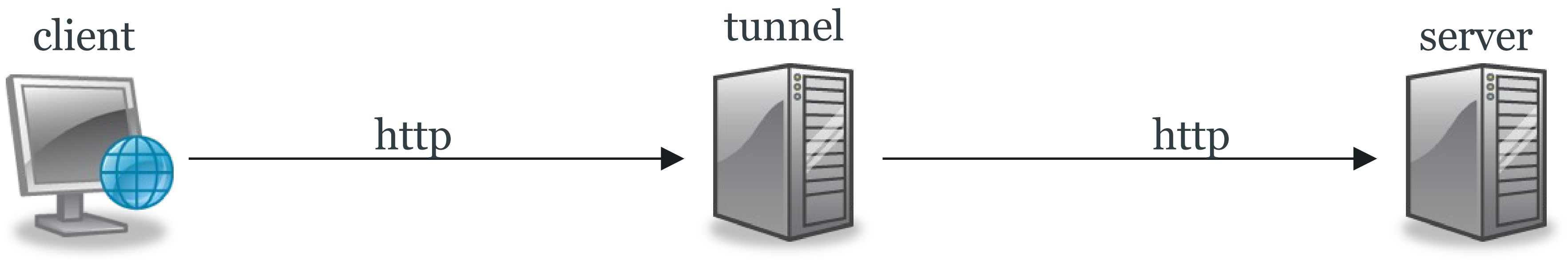
1. receives request
2. rewrites message
3. forwards to server

HTTP Intermediaries: Gateway



1. receives request
2. translates request to server protocol

HTTP Intermediaries: Tunnel



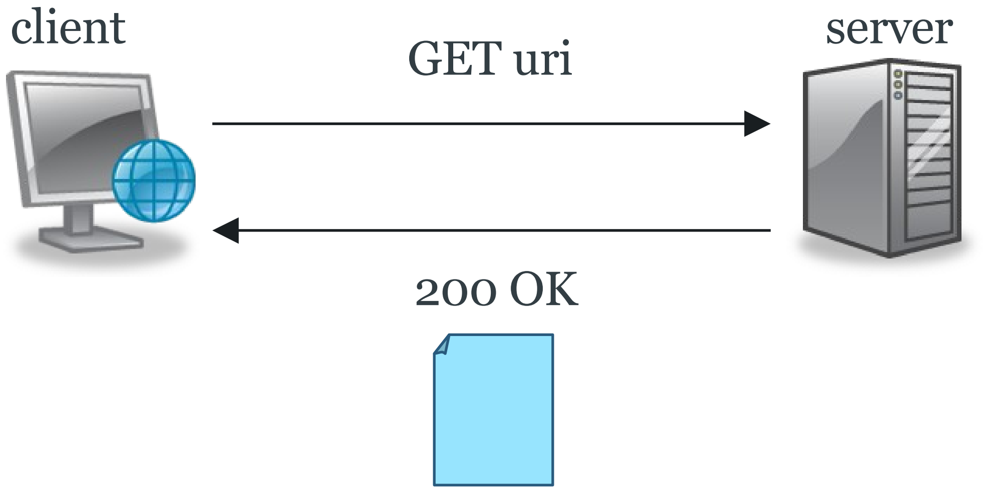
relays between connections
without changing message

UNIVERSITY OF Southampton

HTTP Messages

```
<message>    ::= ( <request> | <response> )
                <header>*
                CRLF
                <body>

<request>::= <method> SP <request-uri> SP
                <http-version> CRLF

<response>    ::= <http-version> SP <status-code> SP
                <reason-phrase> CRLF

<header> ::= <field-name> : <field-value> CRLF

<body>    ::= <sequence of bytes>
```

Typical message exchange



Minimal HTTP/1.1 Exchange

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



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

<html>
<head><title>Acme, Inc Homepage</title></head>
<body><h1>Welcome to Acme!</h1> … </body>
</html>
```

HTTP/1.1 Methods

GET – request a representation of a resource

HEAD – requests the body-less response from a GET request

POST – request that a representation be accepted as a new subordinate of the specified resource

PUT – uploads a representation of the specified resource

DELETE – deletes the specified resource

- (also TRACE, OPTIONS, CONNECT, PATCH)

HTTP/1.1 Request Headers

- Accept: specify desired media type of response
- Accept-Language: specify desired language of response
- Date: date/time at which the message was originated
- Host: host and port number of requested resource
- If-Match: conditional request
- Referer: URI of previously visited resource
- User-Agent: identifier string for Web browser or user agent

HTTP/1.1 Status Codes

1xx – informational message

2xx – success

3xx – redirection

4xx – client error

5xx – server error

200 OK

The request has succeeded.

For a GET request, the response body contains a representation of the specified resource

For a POST request, the response body contains a description of the result of the action

201 Created

The request has been fulfilled and resulted in a new resource being created.

300 Multiple Choices

Multiple representations of the requested resource exist, and the client is provided with negotiation so that it may select a preferred representation

301 Moved Permanently

The requested resource has been assigned a new permanent URI and any future references to this resource SHOULD use one of the returned URIs.

New permanent URI given using the Location: header

302 Found

The requested resource resides temporarily under a different URI. Since the redirection might be altered on occasion, the client SHOULD continue to use the Request-URI for future requests.

Temporary URI given using the Location: header

401 Unauthorized

The request requires user authentication.

The response MUST include a WWW-Authenticate: header field containing a challenge applicable to the requested resource (username/password, for example)

403 Forbidden

The server understood the request, but is refusing to fulfill it. Authorization will not help and the request SHOULD NOT be repeated.

404 Not Found

The server has not found anything matching the Request-URI. No indication is given of whether the condition is temporary or permanent.

405 Method Not Allowed

The method specified in the Request-Line is not allowed for the resource identified by the Request-URI. The response MUST include an Allow: header containing a list of valid methods for the requested resource.

409 Conflict

The request could not be completed due to a conflict with the current state of the resource.

Conflicts are most likely to occur in response to a PUT request. For example, if versioning were being used and the entity being PUT included changes to a resource which conflict with those made by an earlier (third-party) request, the server might use the 409 response to indicate that it can't complete the request.

HTTP/1.1 Response Headers

- Allow: lists methods supported by request URI
- Content-Language: language of representation
- Content-Type: media type of representation
- Content-Length: length in bytes of representation
- Date: date/time at which the message was originated
- Expires: date/time after which response is considered stale
- ETag: identifier for version of resource (message digest)
- Last-Modified: date/time at which representation was last changed

HTTP Content Negotiation

HTTP allows the serving of different representations of a resource based on client preferences

Two areas for negotiation

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

HTTP Content Negotiation Example

```
GET / HTTP/1.1
Host: www.acme.com
Accept: text/html; q=1.0, text/plain; q=0.5
```

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

<html>
<head><title>Acme, Inc Homepage</title></head>
<body><h1>Welcome to Acme!</h1> … </body>
</html>
```

HTTP Content Negotiation Example

```
GET / HTTP/1.1
Host: www.acme.com
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>Acme, Inc Homepage</title></head>
<body><h1>Willkommen zu Acme!</h1> … </body>
</html>
```

HTTP Extensions

WebDAV

HTTP/1.1 still essentially a read-only protocol, *as deployed*

- Web Distributed Authoring and Versioning – HTTP extension
- Most recent version from 1999 – RFC2518

Extra methods:

- PROPFIND – retrieve resource metadata
- PROPPATCH – change/delete resource metadata
- MKCOL – create collection (directory)
- COPY – copy resource
- MOVE – move resource
- LOCK/UNLOCK – lock/release resource (so that others can't change it)

Beyond HTTP/1.1

HTTP Limitations

In order to fetch multiple resources from a server, HTTP/1.0 opens multiple connections to that server

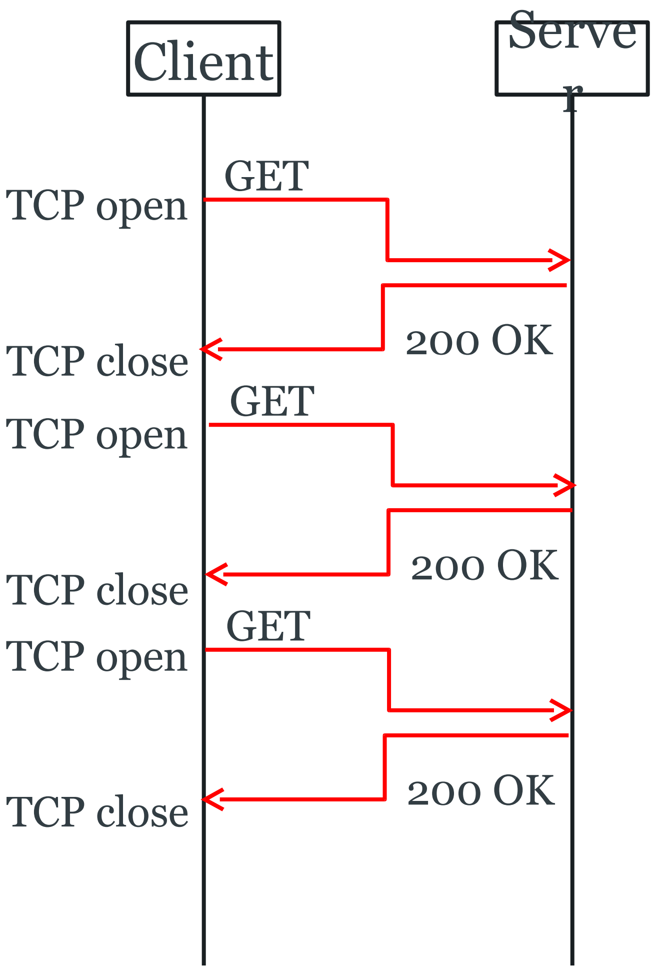
- Extra costs in connection set-up/teardown
- Increased latency if connections are not concurrent

Two partial solutions

- Reuse connections – HTTP Keep-Alive
- Service requests in parallel – HTTP Pipelining

HTTP/1.0 and earlier

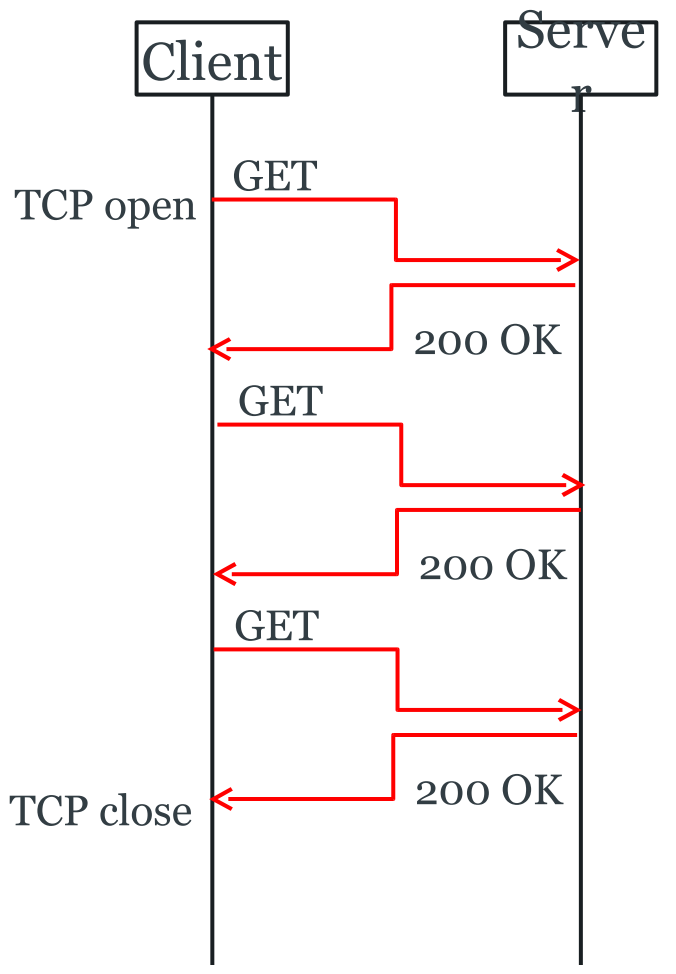
Before HTTP/1.1, each HTTP request used a separate TCP connection



HTTP Keep-Alive

HTTP/1.1 introduced keep-alive

TCP connections reused for multiple HTTP requests



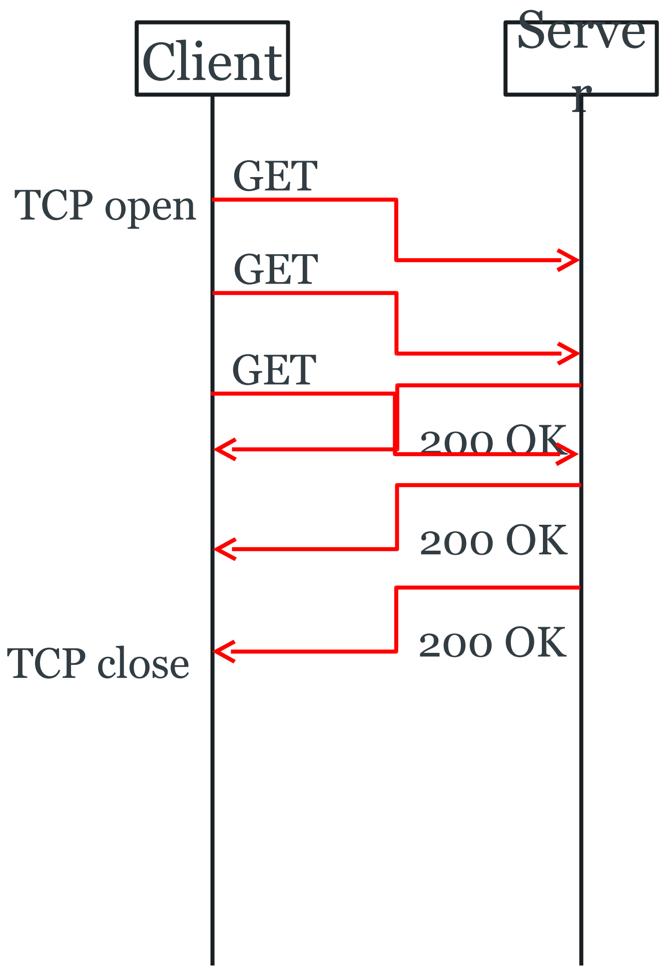
HTTP Pipelining

Also available from HTTP/1.1

Pipelining allows multiple requests to be made without waiting for responses

Server must send responses in same order as received requests

Reduces latency



SPDY

Not an acronym - pronounced ‘speedy’

- Development between Google and Microsoft
- Preserves existing HTTP semantics – SPDY is purely a *framing layer*
- Basis for HTTP/2.0

Offers four improvements over HTTP/1.1:

- Multiplexed requests
- Prioritised requests
- Compressed headers
- Server push

HTTP/2.0 Prioritised Requests

A connection may contain multiple streams (each of which consists of a sequence of frames)

Each stream has a 31-bit identifier

- Odd for client-initiated
- Even for server-initiated

Each stream has another 31-bit integer that expresses its relative priority

- Frames from higher priority streams sent before those from lower priority streams
- Allows asynchronous stream processing (unlike HTTP/1.1 Pipelining)

HTTP/2.0 Compressed Headers

HTTP/1.1 can compress message bodies using gzip or deflate

- Sends headers in plain text

HTTP/2.0 also provides the ability to compress message headers

HTTP/2.0 Push

HTTP/1.1 servers only send messages in response to requests

HTTP/2.0 enables a server to pre-emptively send (or *push*) multiple associated resources to a client in response to a single request.

Further Reading

Hypertext Transfer Protocol – HTTP/1.1

- http://www.w3.org/Protocols/rfc2616/rfc2616.html