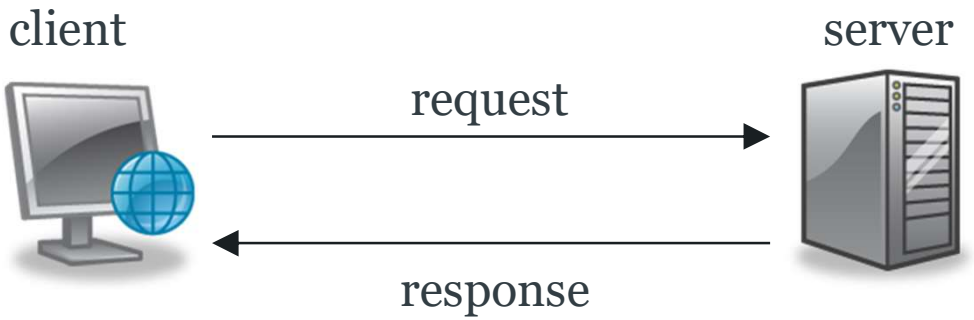
HTTP - Web Protocol

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

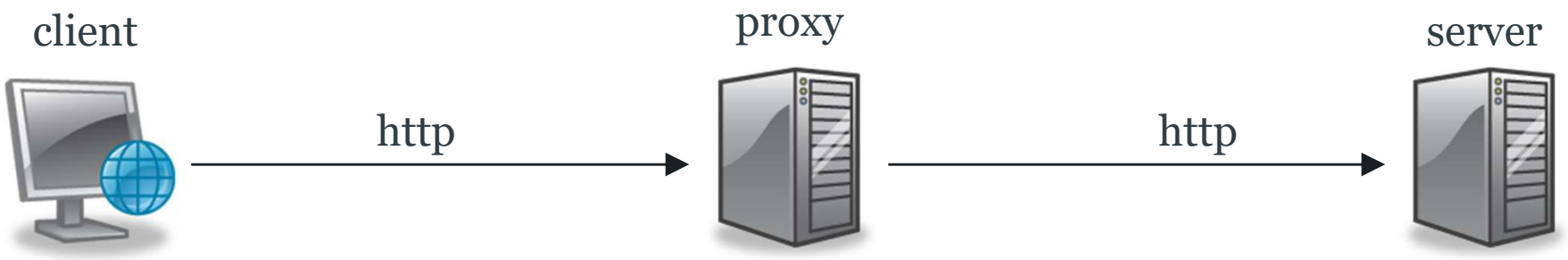


Hypertext Transfer Protocol

Typically a direct connection between client and server
May be intermediaries in the request/response chain

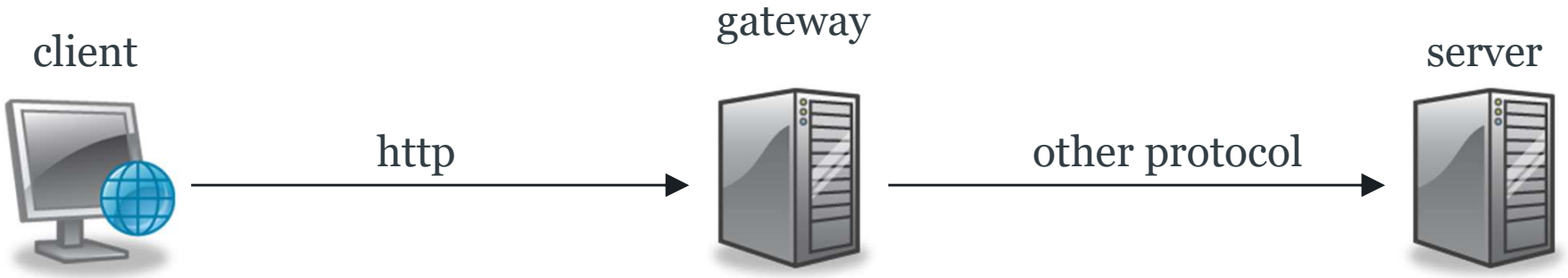
- Proxy
- Gateway
- Tunnel

HTTP Intermediaries: Proxy



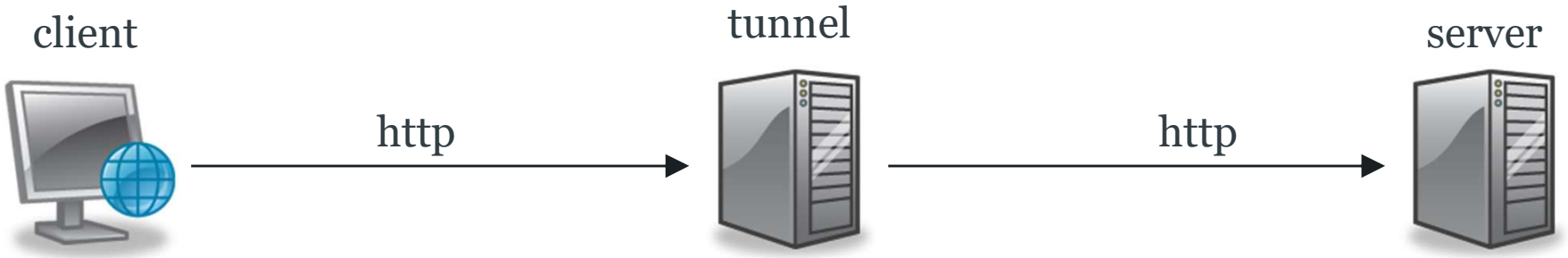
1. receives request
2. rewrites message
3. forwards to server

HTTP Intermediaries: Gateway



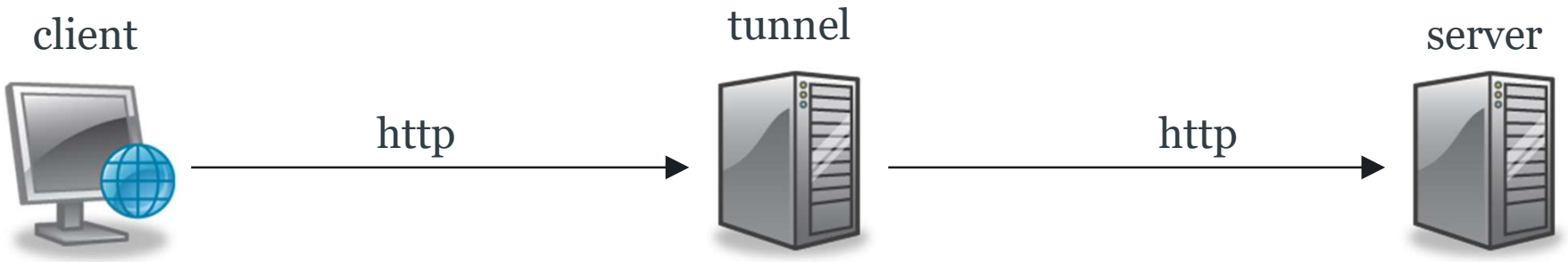
1. receives request
2. translates request to server protocol

HTTP Intermediaries: Tunnel



relays between connections
without changing message

HTTP Intermediaries: Tunnel



relays between connections
without changing message

HTTP Messages

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

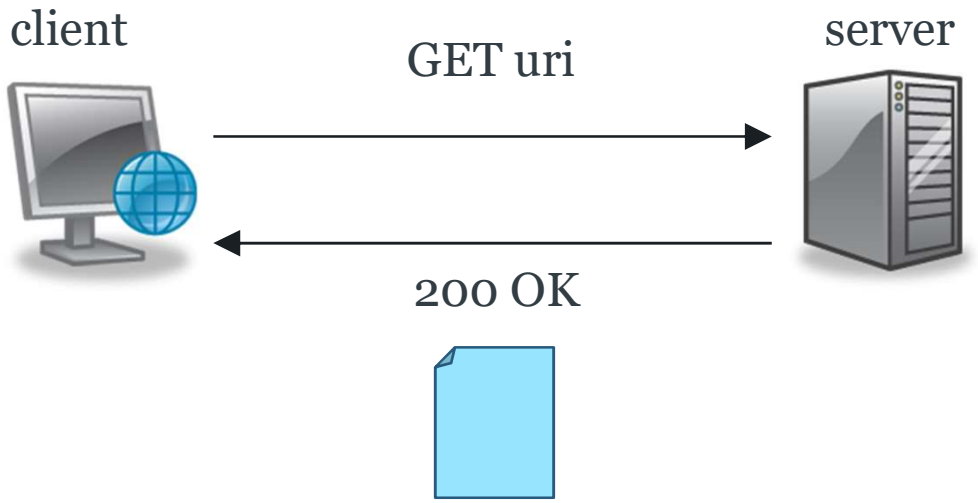
<request> ::= <method> SP <request-uri> SP
<http-version> CRLF

<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
- Referrer: 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

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 Limitations

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

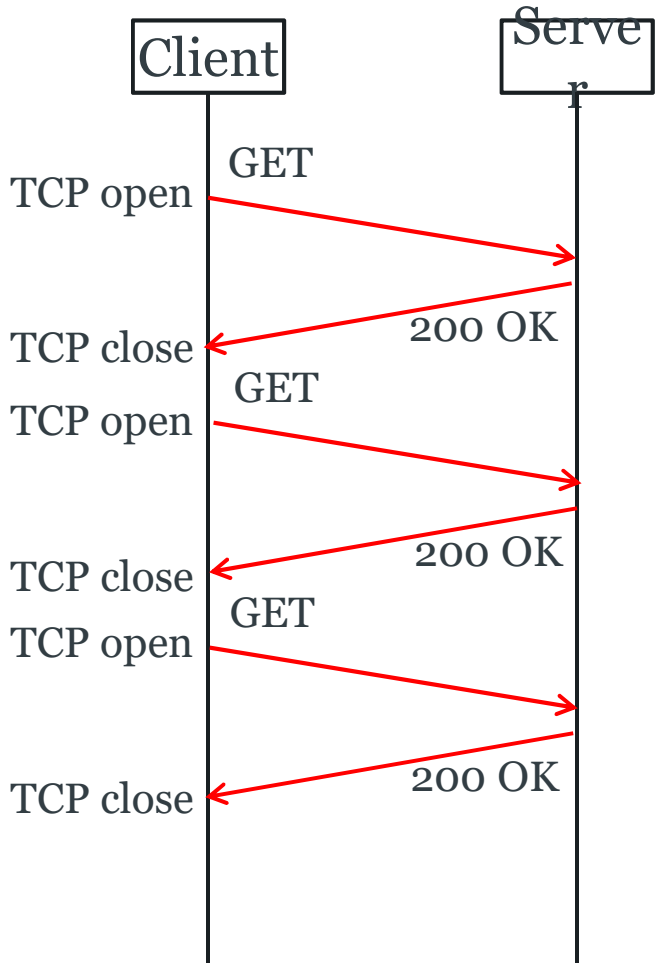
- Extra costs in connection set-up/teardown
- Increased latency if connections are not concurrent

Two partial solutions

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

HTTP/1.0 and earlier

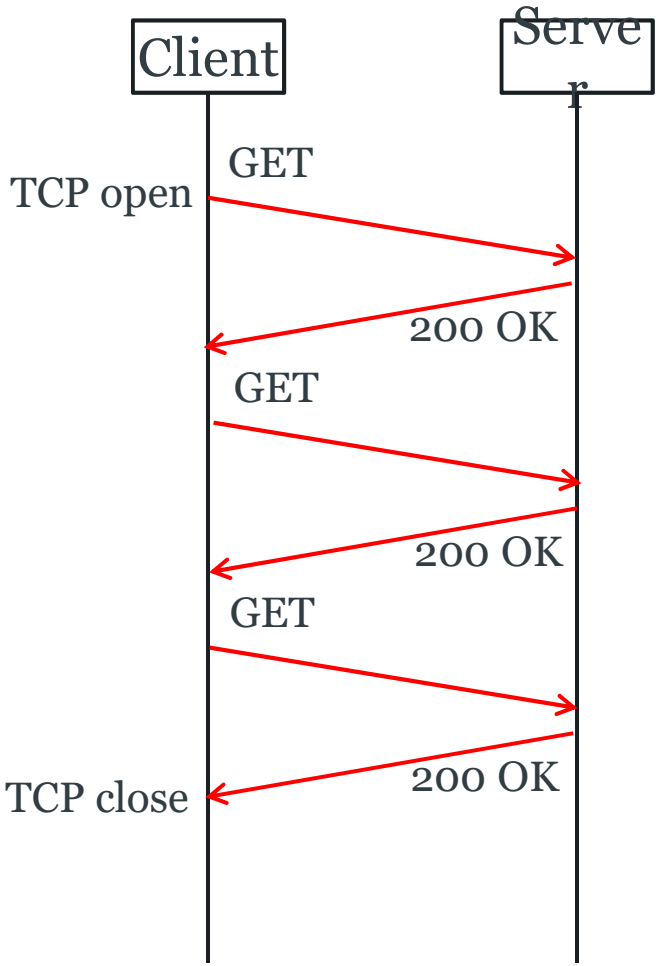
Before HTTP/1.1, each HTTP request used a separate TCP connection



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