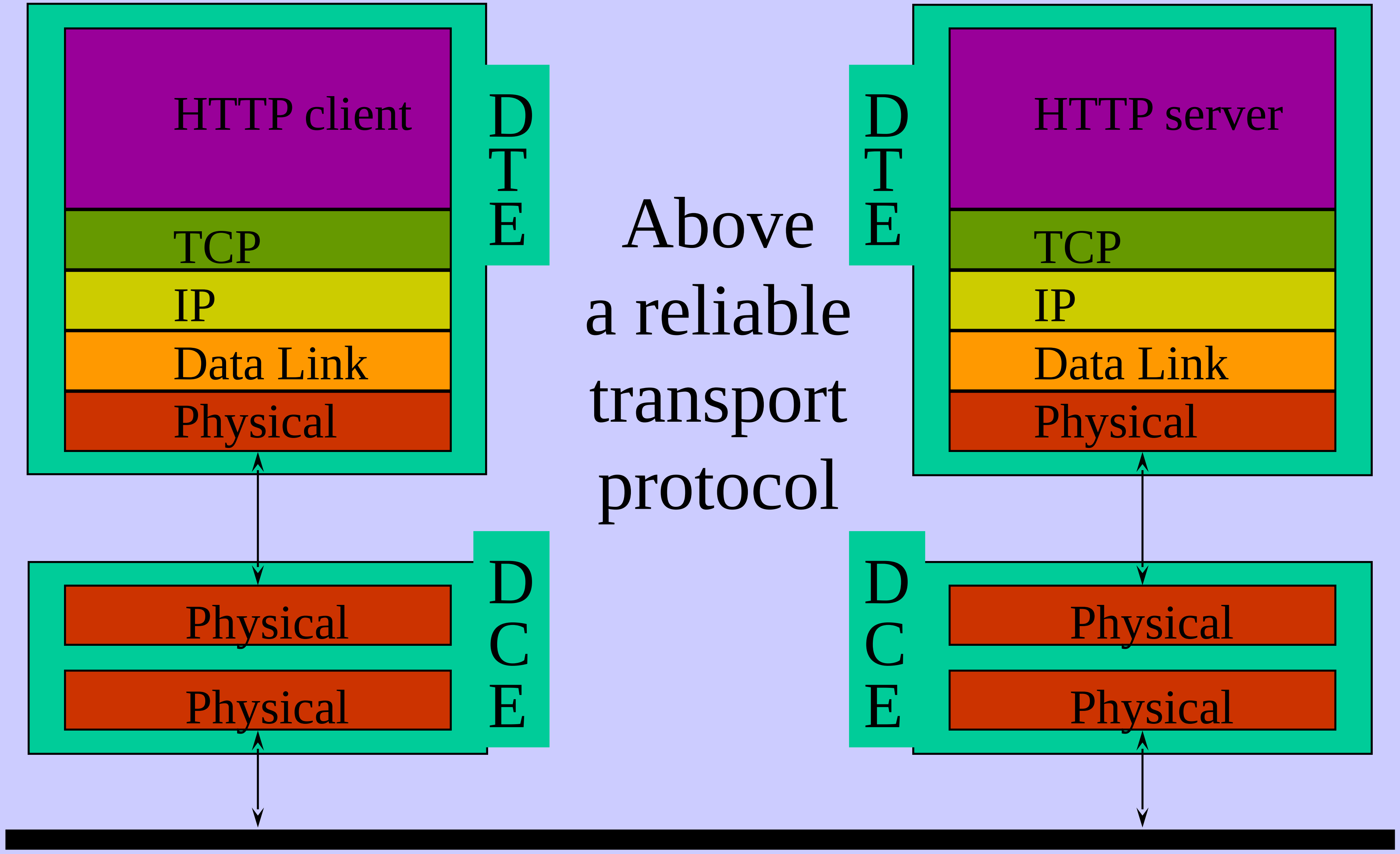
HTTP

- Hypertext Transfer Protocol, in use with the World Wide Web since 1990
- Client request / Server reply
- Client may transmit content
- Server provides public information, but simple forms of access control are possible



Above a reliable transport protocol

Intermediate steps

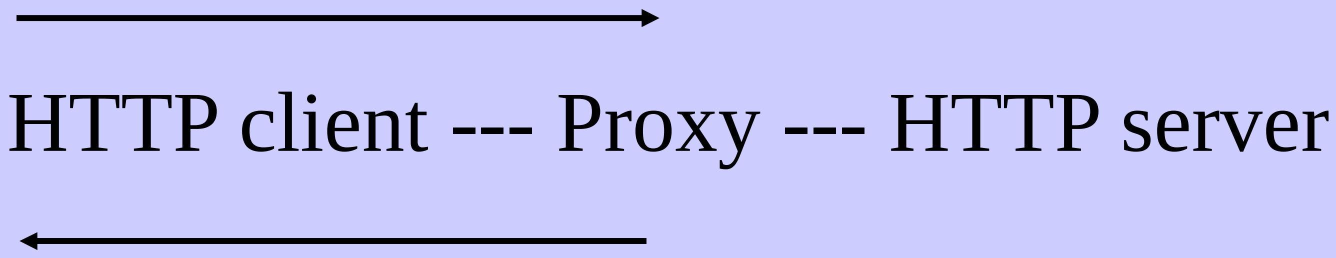
HTTP requests may not be direct and may not really reach their intended, final server.
There are three possibilities:

- Proxies
- Gateways
- Tunnels

Proxies

HTTP requests intented for a particular server are first received by a proxy server. The proxy may either service the request immediately, based on chached information, or forward it to the final server. On the way, back, the information will be passed from the proxy to the client.

Proxies



Gateways

A non-HTTP service may be required: in this case an HTTP request is sent to a gateway, that will translate it to the required protocol and forward it to the corresponding server.

Again, the response will be relayed back to the client under HTTP by the gateway.

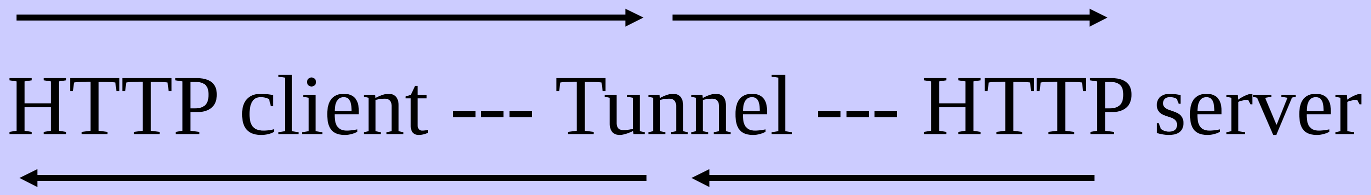
Gateways

$$\begin{tikzcd}
\text{HTTP client} \arrow[r, shift left, "\text{---}"] & \text{Gateway} \arrow[l, shift left] \arrow[r, shift left, "\text{---}"] & \text{--- server (non-HTTP)} \arrow[l, shift left]
\end{tikzcd}$$

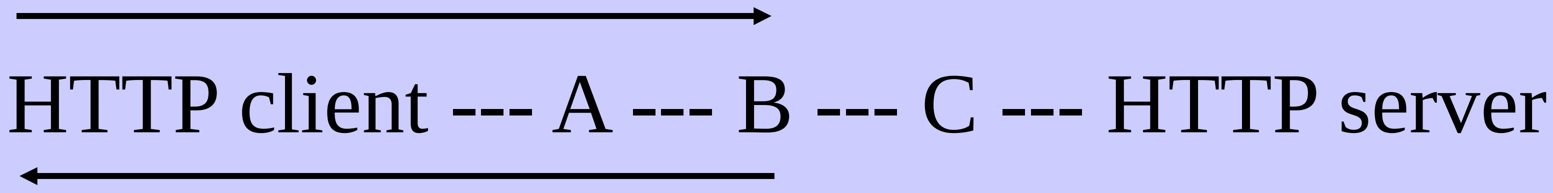
Tunnels

Tunnels receive and inspect the HTTP message, but forward it unchanged to the server. Tunnels may be used for logging or for access control, e.g. in a firewall.

Tunnel



More than one proxy, tunnel or gateway may be used



HTTP requests

Requests refer to a server resource as an Uniform Resource Identifier (URI):

http://host:port/path?query_string

host: server name or IP number
port: service access point (e.g. TCP port)
path: a path to a resource on the server
query: additional client information

HTTP formats

Request:

Method URI Version CRLF

GeneralHdr RequestHdr EntityHdr CRLF

EntityBody

Response:

Version Status-Code Reason-Phrase CRLF

GeneralHdr ResponseHdr EntityHdr CRLF

EntityBody

Request

Method URI Version CRLF
GeneralHdr RequestHdr EntityHdr CRLF
EntityBody

Main methods are Get and Post
EntityBody is optional, used for Post
Headers are optional, and provide extra info
Current Version is HTTP/1.1

Request - Get example

GET http://www.di.unito.it:8887/index.html HTTP/1.0

This will request the information contained in the referenced URI. This will be returned in the server's response HTTP message. A port number of 8887 is used, and "index.html" is the path to a server file.

Request - Post example

```
POST http://www.di.unito.it/cgi/test HTTP/1.0
Content-Length:33
name=user21&phone+number=2123198
```

This will send the entity information (3rd line) to the “test” program on the server, as indicated in the URI. The EntityHdr must be present and must contain the length of the EntityBody in bytes.

Response

Version Status-Code Reason-Phrase CRLF
GeneralHdr ResponseHdr EntityHdr CRLF
EntityBody

Current Version is HTTP/1.1
Status-Code is 3 digits
Reason-Phrase is a comment to Status-Code

Response Status-Code

Success		*Redirection*	
200	*ok*	*301*	*moved permanently*
201	*created*	*302*	*moved temporarily*
202	*accepted*	*304*	*not modified*
204	*no content*		

Client error		*Server error*	
400	*bad request*	*500*	*internal server error*
401	*unauthorized*	*501*	*not implemented*
403	*forbidden*	*502*	*bad gateway*
404	*not found*	*503*	*service unavailable*

Response example

```
HTTP/1.0 200
Content-Length:232
Content-Type:text/html
<HTML> … </HTML>
```

An HTML page is returned from the server in this HTTP response message

Headers

First-line CRLF
GeneralHdr RHdr EntityHdr CRLF
Body

General Header

Response Header

Request Header

Entity Header

General Header

Date:weekday, day-month-year time GMT

(e.g. Date:Mon, 24 Nov 1997 12:22:59 GMT)

Pragma:no-cache

(proxies must forward the request, even when chached information is available)

Request Header

Authorization:credentials (retry after a 401 status)

If-Modified-Since:date (just return a 304 status if not)

Referer:URI (“previous” URI)

User-Agent:client-software (e.g., Netscape, Lynx)

Response Header

Location:absolute-URI (for redirection)

Server:server-software (e.g., Apache, NCSA, etc.)

WWW-Authenticate:challenges

must be included in a response with a 401 (unauthorized) status-code, the challenges select possible authentication schemes.

Entity Header

Content-Encoding:coding (e.g. x-gzip, x-compress)

Content-Type:type/subtype (e.g. text/html)

Content-Length:N number of bytes in the Body

Expires: date (do not cache when expired)

Last-Modified: date

MIME

Multipurpose Internet Mail Extensions

Tipo/Sottotipo

es.

text/html, text/plain, image/gif, application/msword

Basic Authentication - I

The server may require authentication by sending a response with

- an empty Entity
- a 401 (unauthorized) satus-code
- a response header of the type

WWW-Authenticate: Basic realm=“token”

Basic Authentication - II

Either as a consequence of a 401 status, or by its own initiative, the client may provide authentication by including the header

Authorization: Basic basic-cookie

where basic-cookie is a base64 encoding of a userid:password pair.

Basic Authentication - III

If the userid-password is valid for the realm requested in the WWW-Authenticate header, the server will respond with the entity body.

If not, an empty body and a 403 (forbidden) status code is returned.

Basic Authentication - IV

On Unix servers, authentication is requested for all resources in server directories containing a file named “.htaccess”, where access restricions for specific userids are listed.

Passwords are encrypted and saved in a separate file.

Basic Authentication - V

The basic authentication scheme is insecure on normal HTTP servers, because the userid-password basic cookie may be read while in transit and re-used later.

The scheme is valid when a secure transport layer is in place, as it is the case with SSL servers, because HTTP traffic is encrypted.

The common gateway interface (CGI)

An HTTP server may behave as a gateway under the common CGI scheme:

The client request is first processed by the server;

Then the request entity body and some header information is passed to an executable program, corresponding to the request URI;

This program's output is returned to the client as an HTTP response.

CGIs with a method of “GET”

The request must be of the type

GET http://host/cgi/program?query HTTP/1.0

where “program” is executable, and query is additional information from the client, a string of characters.

HTTP servers will normally pass query to program in a null-terminated environment variable called QUERY_STRING. The program’s output will be returned to the client “as is”, preceded by a normal response status-line.

CGIs with a method of “POST”

The request must be of the type

POST http://host/cgi/program HTTP/1.0

Content-Length:N

Entity-Body

HTTP servers will normally pass Entity-Body to program in its standard input, and N in an environment variable called CONTENT-LENGTH. The program’s output will be returned to the client “as is”, preceded by a normal response status-line.

Using CGIs from a Browser

- A normal request, with the CGI program in the URI’s path, and additional information in the URI’s query (server-side information may be computed at the time of the request - see cgip.c)
- A request (either POST or GET) prepared by the Browser on the basis of FORM input.

CGIs and WWW forms (General)

User input to WWW forms comes as a list of name/value pairs (see formtut.html).

The Browser will “URL-encode” such information: spaces are changed to +, special characters are turned to the three characters %xx with xx being the hexadecimal ASCII codes

The name/value pairs are turned into a string such as
name1=value1&name2=value2&…&nameN=valueN

CGIs and WWW forms (GET)

If the FORM is used with a method of GET, the obtained string is sent as a URI query, following the host and the path to the CGI program, in an HTTP request also using a method of GET

The CGI program will obtain the string with the URL-encoded name/value pairs in the QUERY-STRING environment variable. The last character in the string is a 0, content length is not needed.

CGIs and WWW forms (POST)

If the FORM is used with a method of POST, the obtained string is sent as the request’s entity body, with a corresponding Content-Length header.

The CGI program will obtain the string with the URL-encoded name/value pairs in its standard input. A null-character is not appended to the string, and content length is needed.

CGIs and WWW forms (decoding)

The CGI programs will then need to URL-decode the user input, and, in the case of forms, split it into separate name/value pairs.

The output must include possible content-type headers, and an entity body that will be returned in the server’s response.

See post_query.c, query.c, util.c

Keep-alive / HTTP/1.1

one TCP/connection for many HTTP requests

-> performance

Cookies

```
HTTP/1.1 302 Object moved
Location: /redazione/default.asp
Content-Type: text/html
Set-Cookie:
ASPSESSIONIDGGQQGHFF=GIIJAMHCHKAPAFPDKIPBMOJD;
path=/

<head><title>Object moved</title></head>
<body><h1>Object Moved</h1>
</body>
```

LOG dei server HTTP

```
213.140.17.110 - -
[02/Dec/2004:17:42:56 +0100]
"GET /favicon.ico HTTP/1.1" 200 3262
```

Referer
Date
Indirizzi IP
Cookies

-> Web analytics / Web mining: tracciare l’utente, capire qual è la entry page, la exit page, durata delle visite, contare il numero di visitatori.