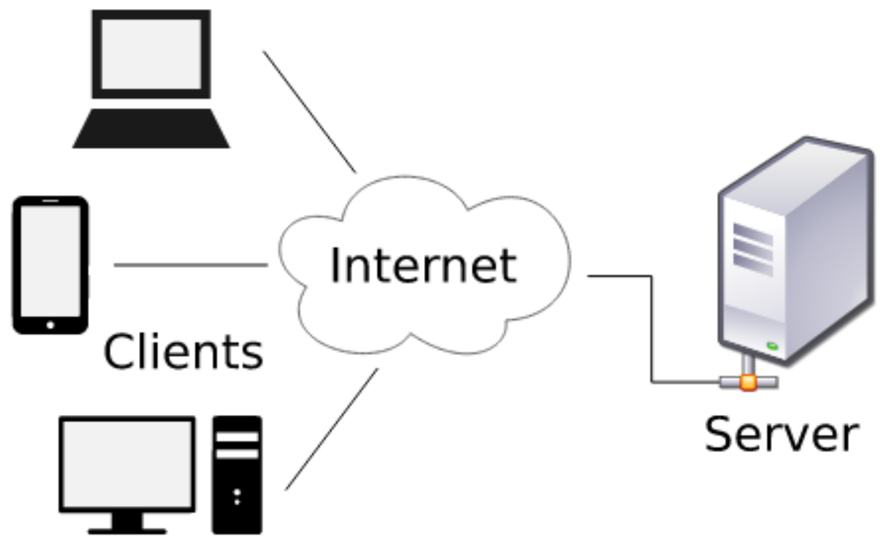
Sockets API and HTTP 1.0

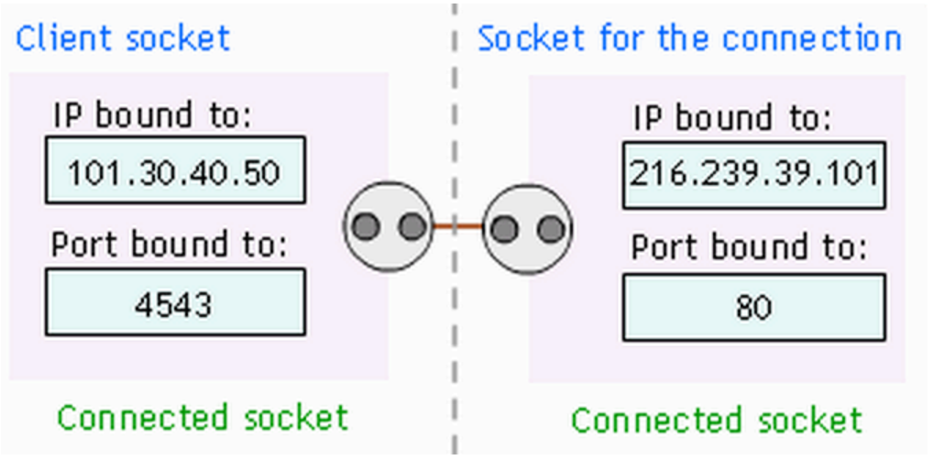
Prepared by Ruchir Khaitan
Edited by Jae Woo Lee

The client / server model



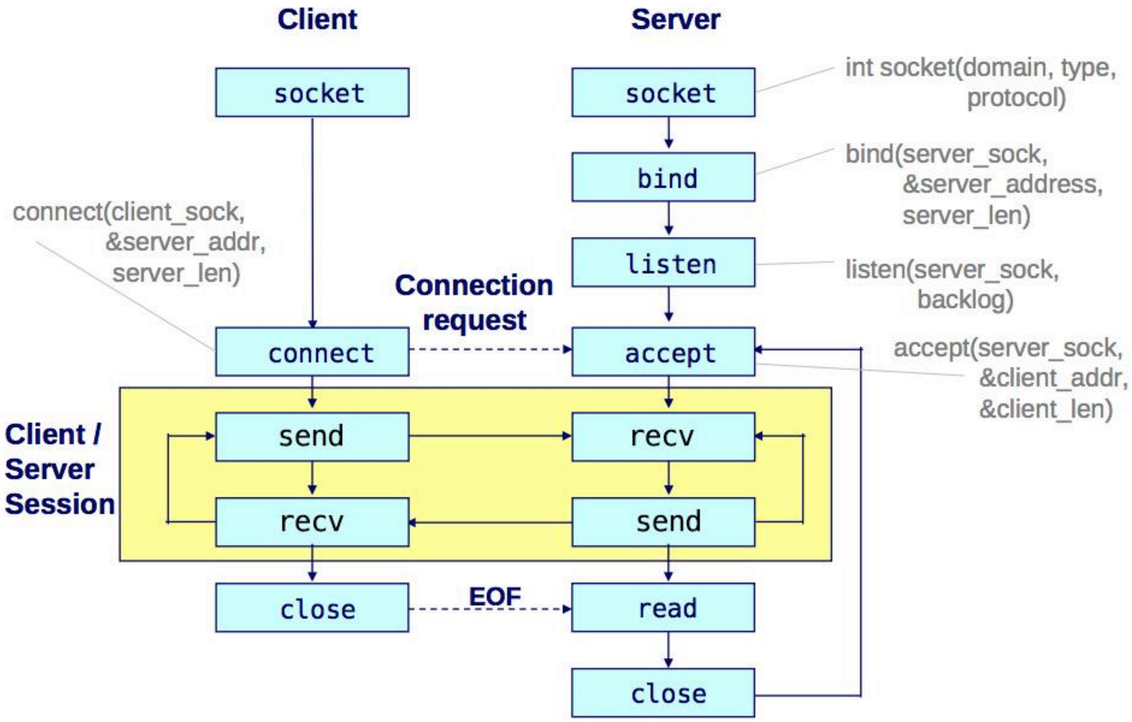
- Server waits for incoming requests over the network from clients
- e.g. Web browser & web server
- C programs use Sockets API

What is a socket?



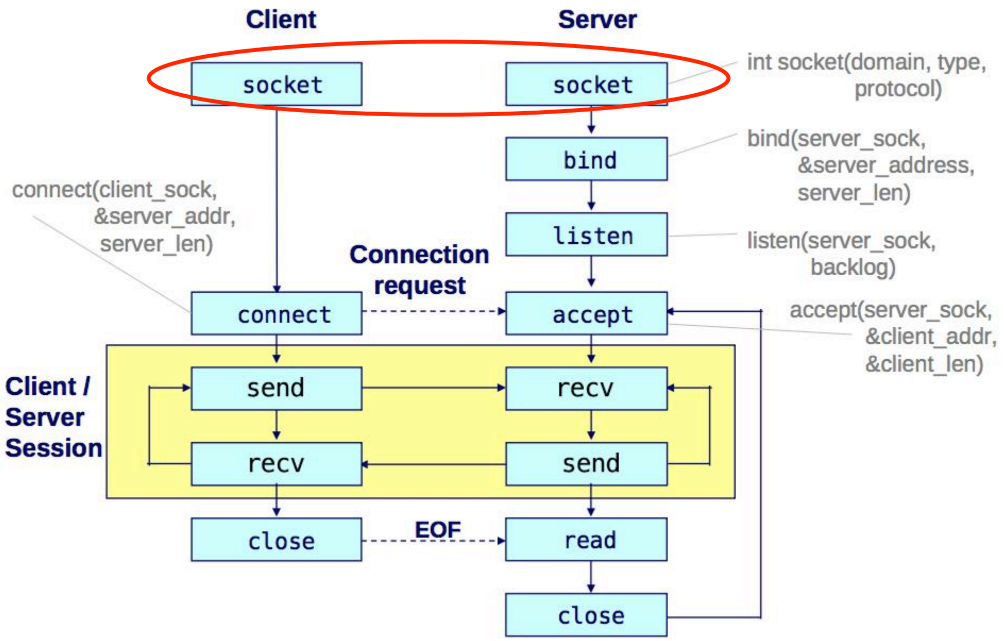
- End-point for inter-process communication over TCP/IP network
- Socket is bound to an IP address and a port number

Sockets API Summary


Client
Server
socket
socket
int socket(domain, type, protocol)
bind
bind(server_sock, &server_address, server_len)
listen
listen(server_sock, backlog)
connect(client_sock, &server_addr, server_len)
Connection request
connect
accept
accept(server_sock, &client_addr, &client_len)
Client / Server Session
send
recv
recv
send
close
EOF
read
close

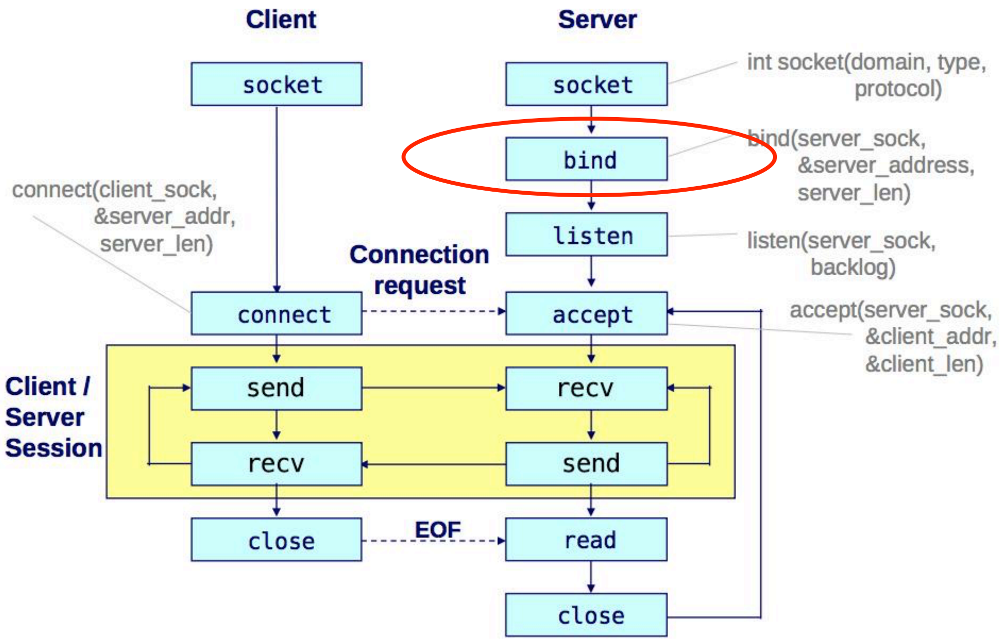
Source: https://roxanageambasu.github.io/ds-class//assets/lectures/lecture4.pdf

socket()


Client
Server
socket
socket
int socket(domain, type, protocol)
bind
bind(server_sock, &server_address, server_len)
listen
listen(server_sock, backlog)
connect(client_sock, &server_addr, server_len)
Connection request
connect
accept
accept(server_sock, &client_addr, &client_len)
Client / Server Session
send
recv
recv
send
close
EOF
read
close

- Called by both client and server
- On the server, a listening socket is created first using socket() – a *connected* socket will be created later by accept()

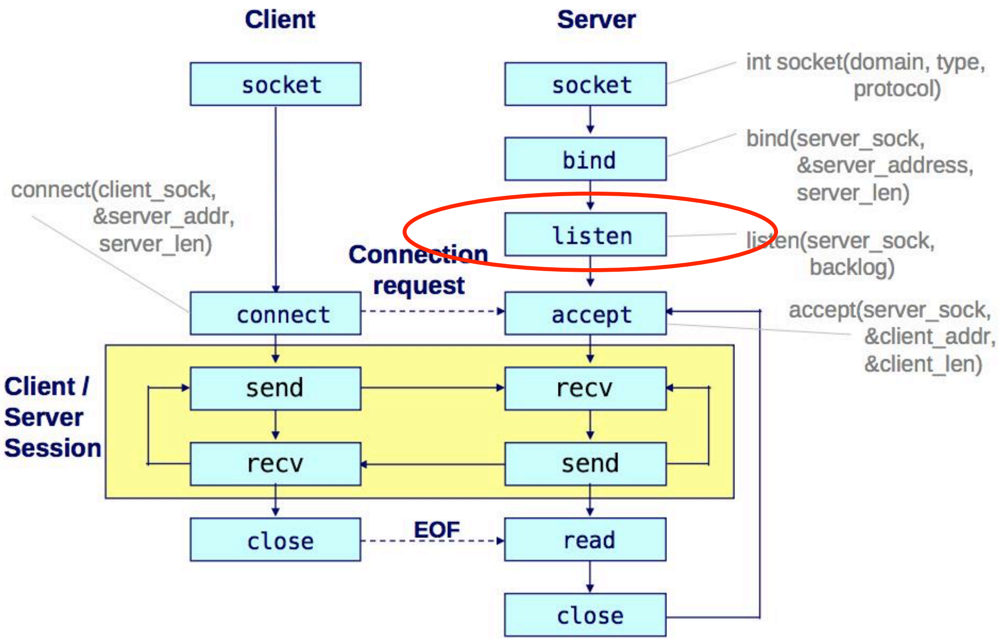
bind()



Client
Server
socket
socket
int socket(domain, type, protocol)
bind
bind(server_sock, &server_address, server_len)
connect(client_sock, &server_addr, server_len)
listen
listen(server_sock, backlog)
Connection request
connect
accept
accept(server_sock, &client_addr, &client_len)
Client / Server Session
send
recv
recv
send
close
EOF
read
close

- Usually called only by server
- Binds the listening socket to a specific port that should be known to the client

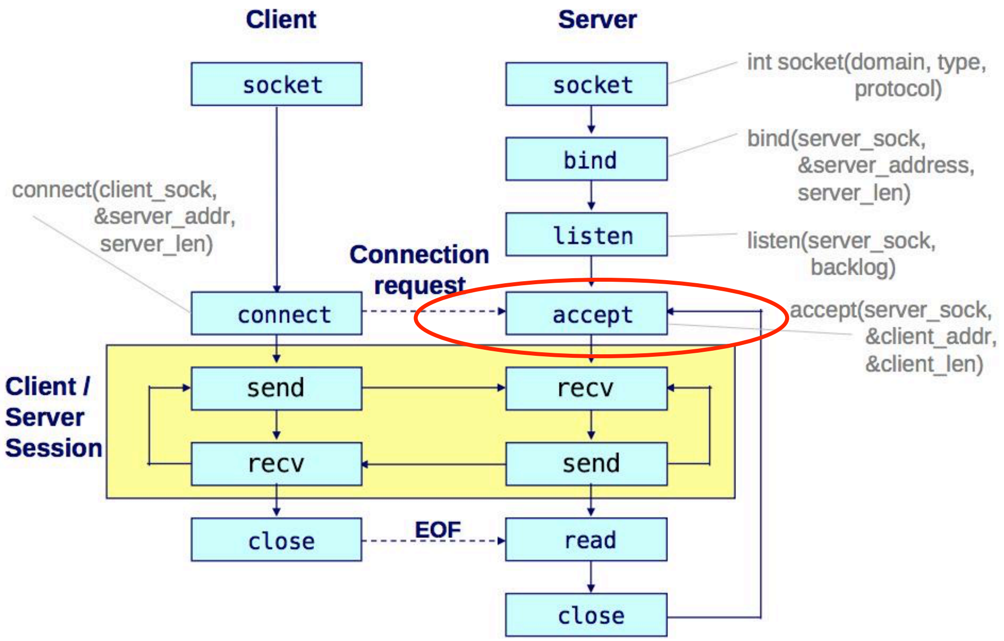
listen()



Client
Server
socket
socket
int socket(domain, type, protocol)
bind
bind(server_sock, &server_address, server_len)
listen
listen(server_sock, backlog)
connect(client_sock, &server_addr, server_len)
Connection request
connect
accept
accept(server_sock, &client_addr, &client_len)
Client / Server Session
send
recv
recv
send
close
EOF
read
close

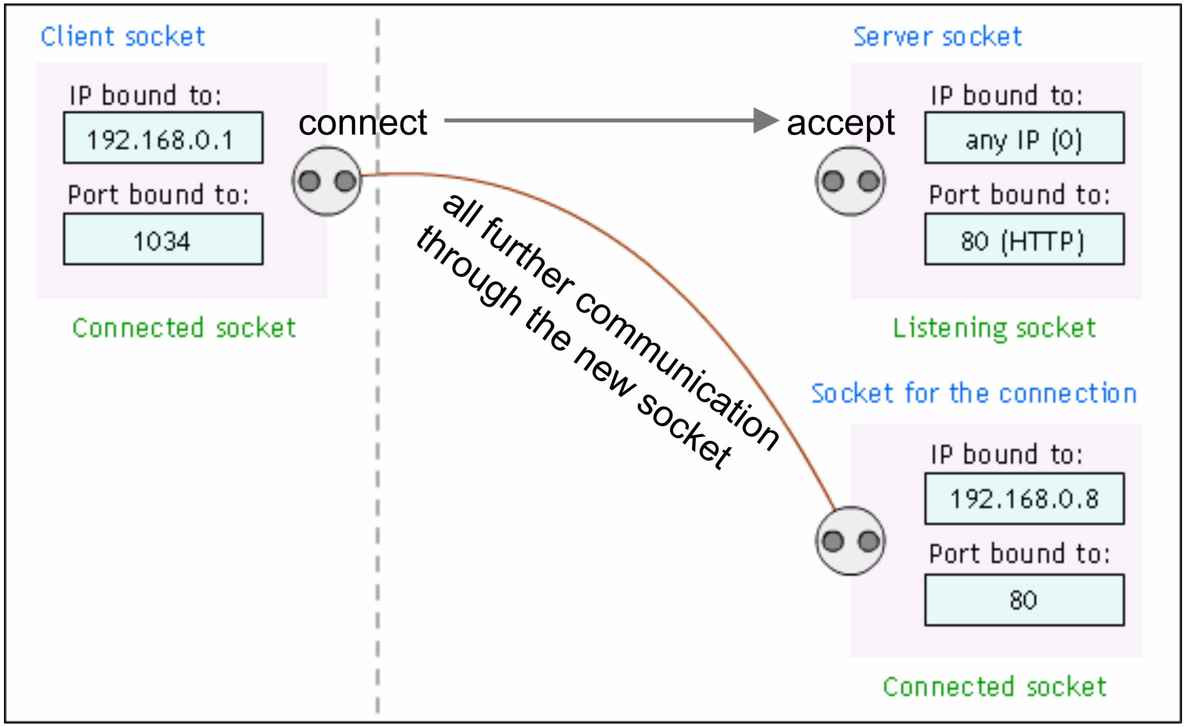
- Called only by server
- Sets up the listening socket to accept connections

accept()


Client
Server
socket
socket
int socket(domain, type, protocol)
bind
bind(server_sock, &server_address, server_len)
listen
listen(server_sock, backlog)
connect(client_sock, &server_addr, server_len)
Connection request
connect
accept
accept(server_sock, &client_addr, &client_len)
Client / Server Session
send
recv
recv
send
close
EOF
read
close

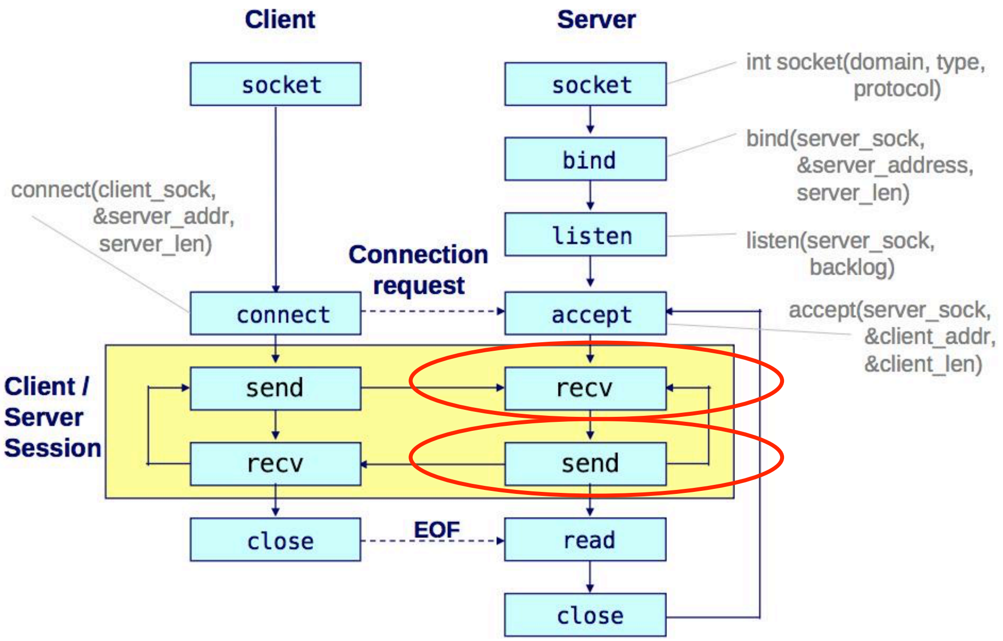
- Called only by server
- By default blocks until a connection request arrives
- Creates and returns a new socket for each new client

Listening socket vs. Connected socket



Source: http://www.madwizard.org/programming/tutorials/netcpp/3

send() and recv()


Client
Server
socket
socket
int socket(domain, type, protocol)
bind
bind(server_sock, &server_address, server_len)
connect(client_sock, &server_addr, server_len)
listen
listen(server_sock, backlog)
Connection request
connect
accept
accept(server_sock, &client_addr, &client_len)
Client / Server Session
send
recv
recv
send
close
EOF
read
close

- Called by both client and server
- Reads and writes to the other side
- Message boundaries may not be preserved

HTTP 1.0

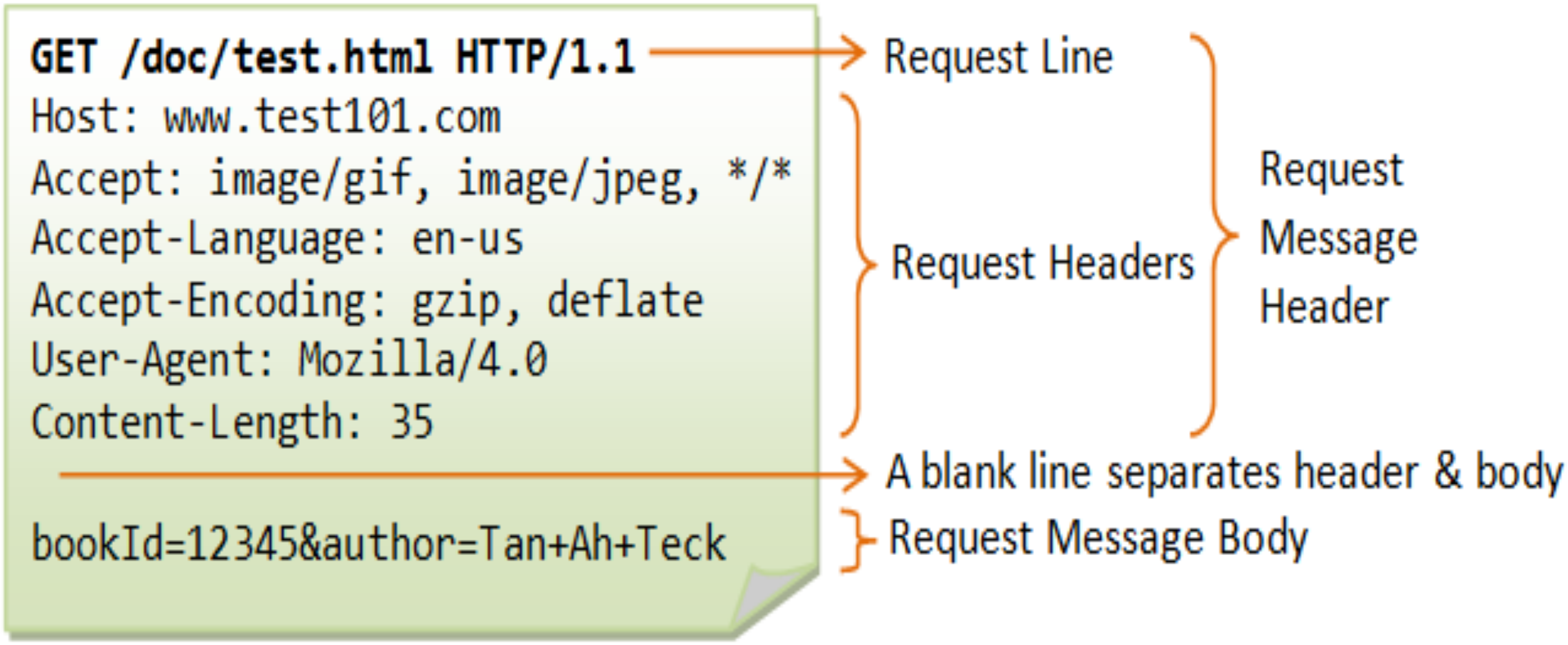
- Client sends a **HTTP request** for a resource on the server (ex. a file on the server)

- The server sends a **HTTP response**

HTTP request

- First line: method, request-URI, version
    - Ex: “GET /index.html HTTP/1.0\r\n”
- Followed by 0 or more headers:
    - Ex: “Host: www.google.com\r\n”
- Followed by an empty line
    - “\r\n”

HTTP request example



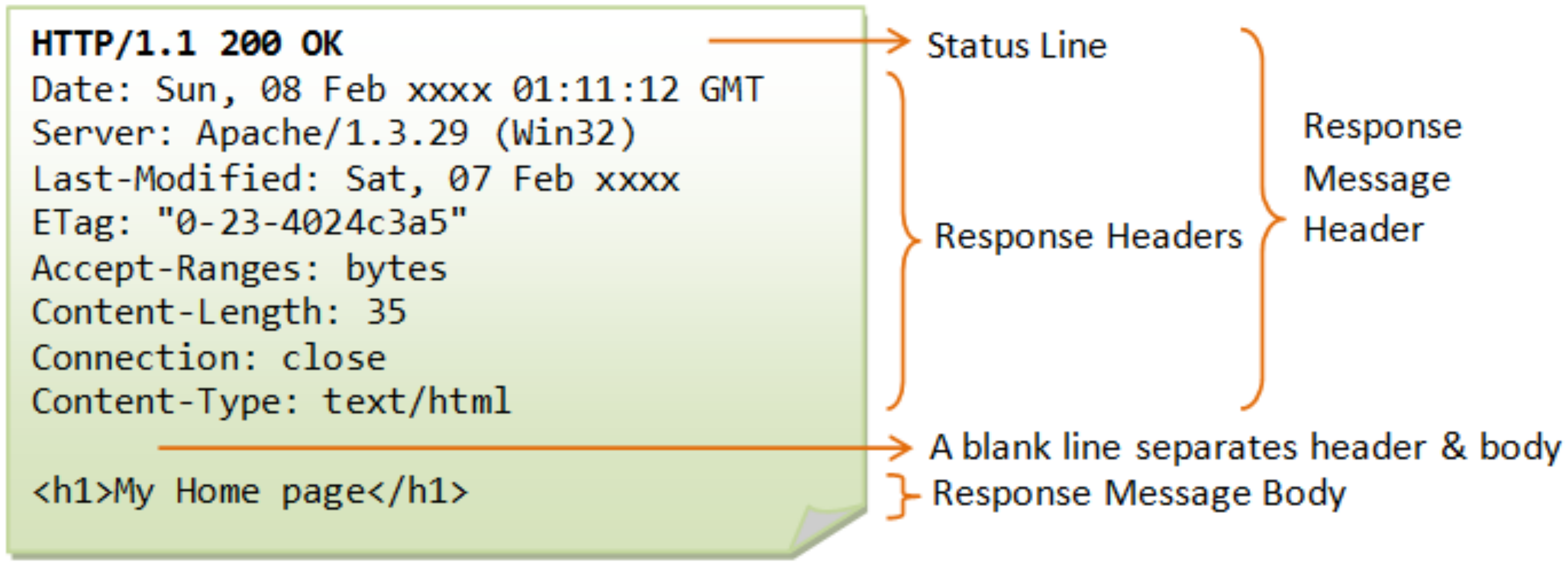
```
GET /doc/test.html HTTP/1.1
Host: www.test101.com
Accept: image/gif, image/jpeg, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0
Content-Length: 35

bookId=12345&author=Tan+Ah+Teck
```

HTTP response

- First line: response status
  - Success: HTTP/1.0 200 OK\r\n
  - Failure: HTTP/1.0 404 Not Found\r\n
- Followed by 0 or more response headers
- Followed by a blank line
- Followed by the content of the response
  - For example, an image file or an HTML file

HTTP response example



```
HTTP/1.1 200 OK
Date: Sun, 08 Feb xxxx 01:11:12 GMT
Server: Apache/1.3.29 (Win32)
Last-Modified: Sat, 07 Feb xxxx
ETag: "0-23-4024c3a5"
Accept-Ranges: bytes
Content-Length: 35
Connection: close
Content-Type: text/html

<h1>My Home page</h1>
```