Application layer and HTTP

Agenda

- Application layer architectures
- Application layer protocols
- HTTP
    - Properties of HTTP
    - Structure of an HTTP request and response

Key terms

Application layer

> The application layer contains the communications protocols and interface methods used in process-to-process communications across an Internet Protocol (IP) computer network.

Client-server model

> is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients.

Peer-to-peer model

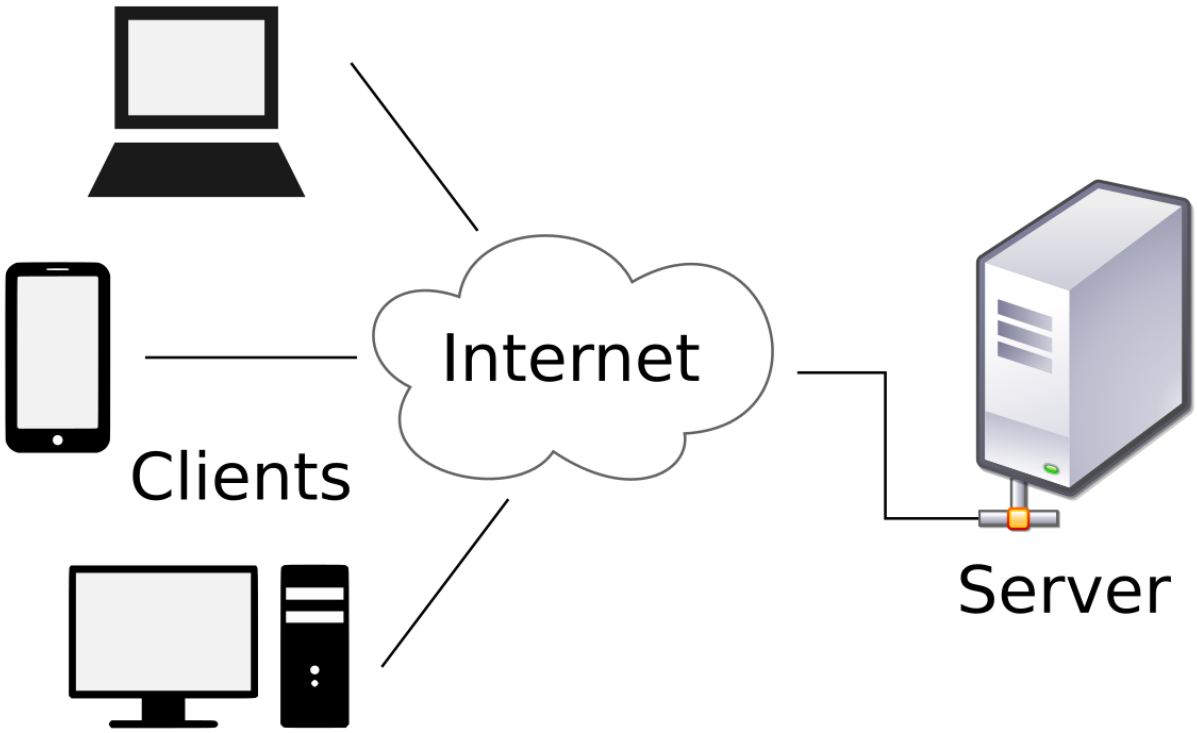
> is a distributed application architecture that partitions tasks or workloads between peers. Peers are equally privileged, equipotent participants in the network

Application layer architectures

There are different networking models for how to connect computers over a network. Computers that request information are called clients and computers that provide information are servers. But the client and server relationship can be organised in different ways. The two most commonly used architectures are:

Client-server

The client-server model is the relationship between two computers in which one, the client, makes a service request from another, the server. The key point about a client-server model is that the client is dependent on the server to provide and manage the information.



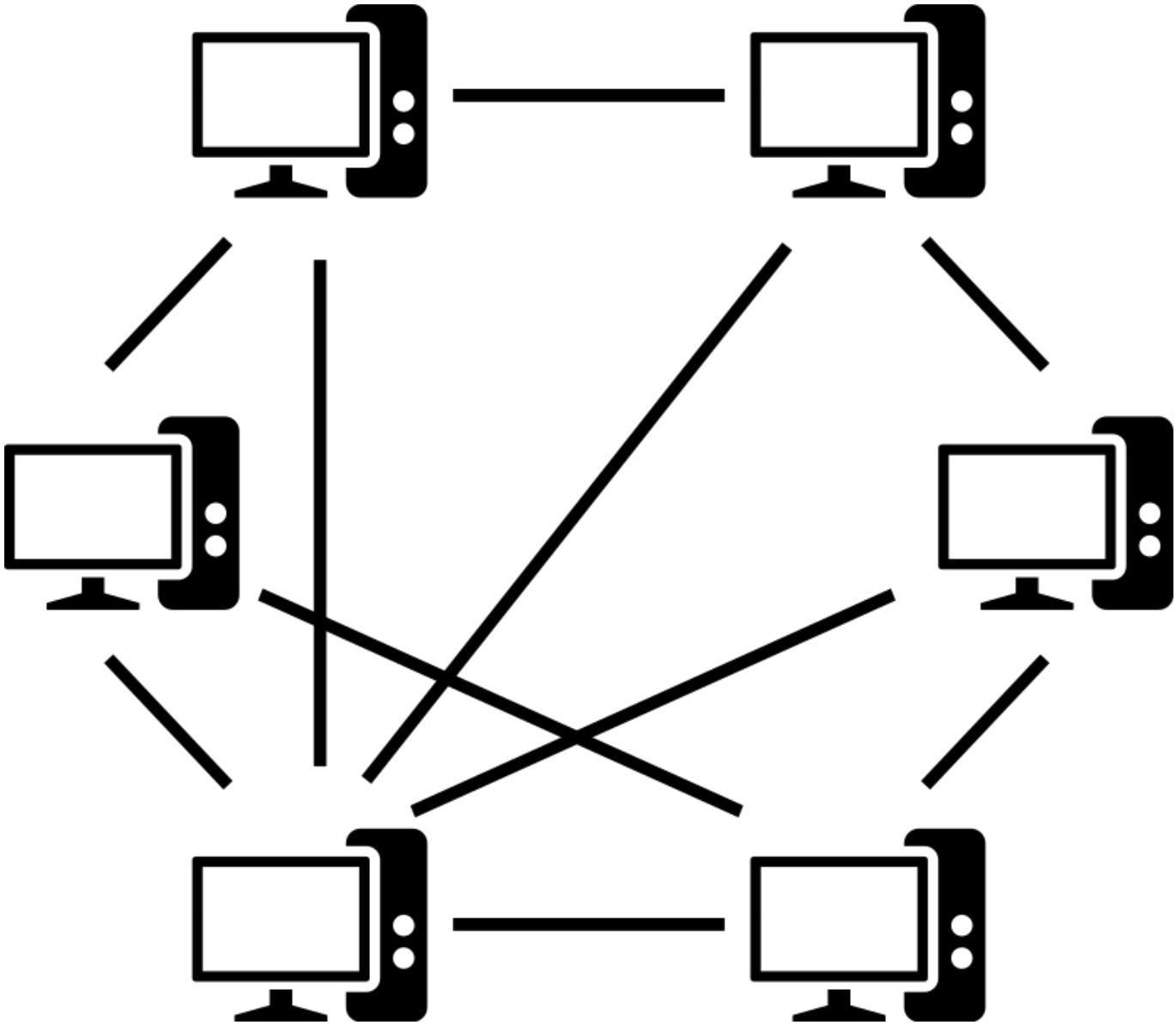
> The client is responsible for consuming data either from the user or from the server

> The server is responsible for generating data.

> For example, websites are stored on web servers. A web browser is the client which makes a request to the server, and the server sends the website to the browser.

Peer-to-peer

In the client-server model, many users trying to access a large file, such as a film, would put strain on one server. In the peer-to-peer model, many users on the network could store the same file. Each computer can then send sections of the file, sharing the workload. Each client can download and share files with other users.



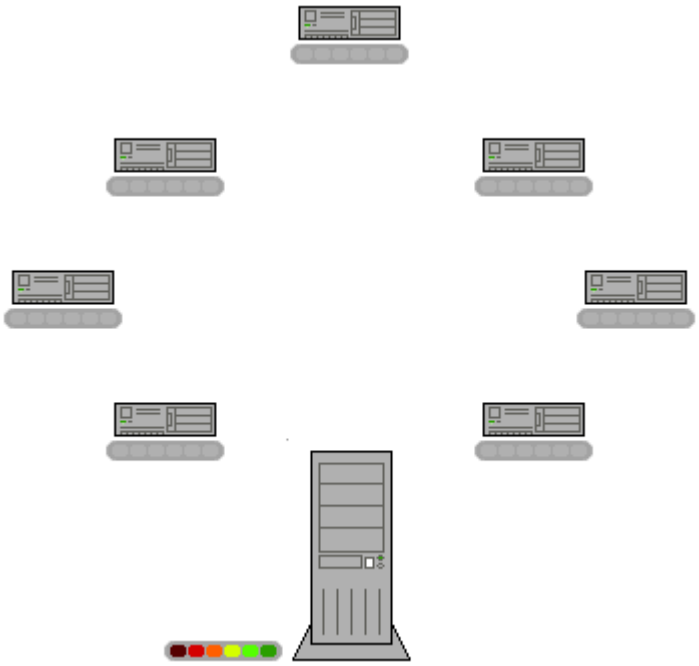
> In a P2P network, no single provider is responsible for being the server. Each computer stores files and acts as a server. Each computer has equal responsibility for providing data

Some applications of the peer-to-peer model are:

- Content delivery - Torrents
- File sharing - Torrents, DC++
- Cryptocurrency - Bitcoin, Ethereum, Peercoin, etc.

BitTorrent

The BitTorrent protocol can be used to reduce the server and network impact of distributing large files. Rather than downloading a file from a single source server, the BitTorrent protocol allows users to join a "swarm" of hosts to upload and download from each other simultaneously. Using the BitTorrent protocol, several basic computers, such as home computers, can replace large servers while efficiently distributing files to many recipients.



- To download a file, the user needs to have a BitTorrent client installed on their computer.
- A user downloads a `.torrent` file which contains a list of trackers.
- The "tracker" server keeps track of where file copies reside on peer machines, which ones are available at time of the client request, and helps coordinate efficient transmission and reassembly of the copied file.
- The file being distributed is divided into segments called pieces. As each peer receives a new piece of the file, it becomes a source (of that piece) for other peers, relieving the original seed from having to send that piece to every computer or user wishing a copy.
- When a user initiates the download, the client gets a list of peers from the tracker.
- Pieces are typically downloaded non-sequentially, and are rearranged into the correct order by the BitTorrent client, which monitors which pieces it needs, and which pieces it has and can upload to other peers

Differences between client-server and peer-to-peer

	Client-server	**Peer-to-peer**
Security	The server controls security of the network.	No central control over security.
Management	The server controls security of the network.	No central control over security.
Dependency	The server manages the network. Needs a dedicated team of people to manage the server.	No central control over the network. Anyone can set up.
Performance	The server can be upgraded to be made more powerful to cope with high demand.	If machines on the network are slow they will slow down other machines.
Backups	Data is all backed up on the main server.	Each computer has to be backed up. Data can easily be deleted by users.

HTTP

The Hypertext Transfer Protocol (HTTP) is the foundation of the World Wide Web, and is used to load web pages using hypertext links. HTTP is an application layer protocol designed to transfer information between networked devices and runs on top of other layers of the network protocol stack. A typical flow over HTTP involves a client machine making a request to a server, which then sends a response message.

Properties of HTTP

1. Client-server protocol.
2. All HTTP versions till 2.0 use TCP as the transport layer protocol. HTTP/3 now uses QUIC.
3. Stateless protocol
4. HTTP responses are considered as objects that are identified by a unique URI.

Stateless protocol

A stateless protocol is a communication protocol in which the receiver must not retain session state from previous requests. The sender transfers relevant session state to the receiver in such a way that every request can be understood in isolation, that is without reference to session state from previous requests retained by the receiver.

In contrast, a stateful protocol is a communication protocol in which the receiver may retain session state from previous requests.

Stateless protocols improve the properties of visibility, reliability, and scalability. Visibility is improved because a monitoring system does not have to look beyond a single request in order to determine its full nature. Reliability is improved because it eases the task of recovering from partial failures. Scalability is improved because not having to store session state between requests allows the server to quickly free resources and further simplifies implementation.

The disadvantage of stateless protocols is that they may decrease network performance by increasing the repetitive data sent in a series of requests, since that data cannot be left on the server and reused.

Persistent connections

HTTP connections can be of two types:

1. Non-persistent - The connection is closed after each request.
2. Persistent - The connection is kept open for multiple requests.

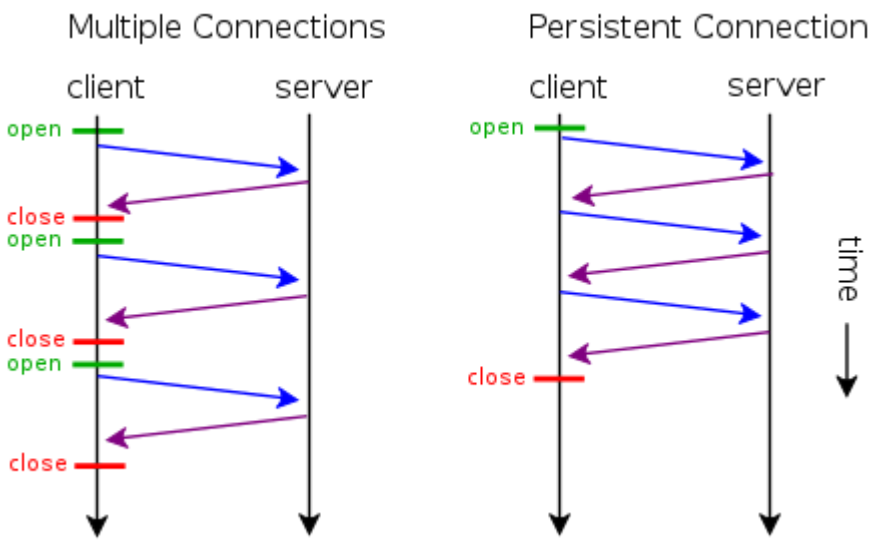
Under HTTP 1.0, connections should always be closed by the server after sending the response. If the client supports `keep-alive`, it adds a header to the request:

```
Connection: keep-alive
```

When the server receives this request and generates a response, if it supports keep-alive then it also adds the same above header to the response. Following this, the connection is not dropped, but is instead kept open. When the client sends another request, it uses the same connection.

This will continue until either the client or the server decides that the conversation is over and in this case they omit the "Connection:" header from the last message sent or, better, they add the keyword "close" to

it



Advantages of persistent connections

- Reduced latency in subsequent requests (no handshaking and no slow start).
- Reduced CPU usage and round-trips because of fewer new connections and TLS handshakes.
- Enables HTTP pipelining of requests and responses.
- Reduced network congestion (fewer TCP connections).
- Errors can be reported without the penalty of closing the TCP connection.

What's in an HTTP request?

A typical HTTP request contains:

- HTTP version
- URL
- HTTP method
- HTTP request headers
- Optional HTTP body.

HTTP methods

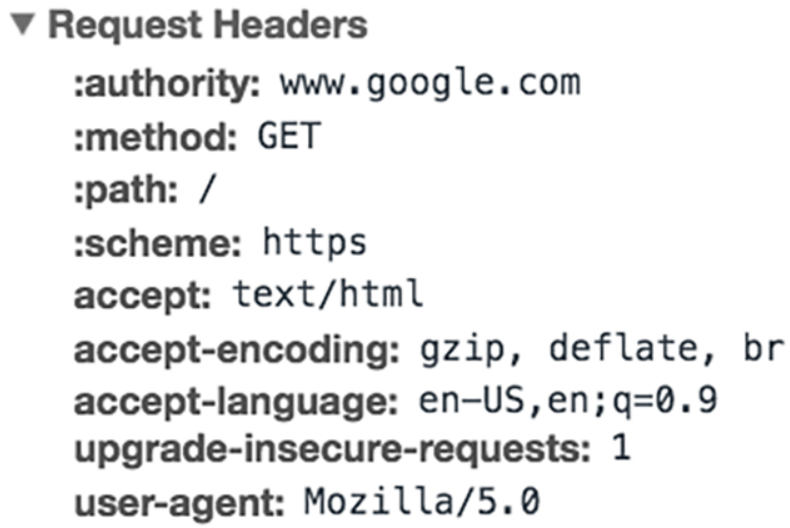
HTTP defines a set of request methods to indicate the desired action to be performed for a given resource. Although they can also be nouns, these request methods are sometimes referred to as HTTP verbs. Each of them implements a different semantic, but some common features are shared by a group of them.

Common methods are:

- GET - Requests using GET should only retrieve data.
- POST - The POST method submits an entity to the specified resource, often causing a change in state or side effects on the server.
- PUT - The PUT method replaces all current representations of the target resource with the request payload.
- PATCH - The PATCH method applies partial modifications to a resource.
- DELETE - The DELETE method deletes the specified resource.

HTTP request headers

HTTP headers contain text information stored in key-value pairs, and they are included in every HTTP request (and response, more on that later). These headers communicate core information, such as what browser the client is using what data is being requested.



Some common headers are:

- Host - The host header specifies the domain name or IP address of the server receiving the request.
- Referer - The Referer header specifies the address of the previous web page from which a link to the currently requested page was followed.
- User-Agent - The User-Agent header specifies the name of the user agent originating the request.
- Accept - The Accept header is used to specify what content types the client is willing to accept.
- Connection - The Connection header is used to control whether the connection should be closed after the request is complete.

See a complete list of HTTP headers here.

What's in an HTTP response?

An HTTP response is what web clients (often browsers) receive from an Internet server in answer to an HTTP request. These responses communicate valuable information based on what was asked for in the HTTP request.

A typical HTTP response contains:

- HTTP status code
- HTTP response headers
- Optional HTTP body.

HTTP status codes

HTTP status codes are 3-digit codes most often used to indicate whether an HTTP request has been successfully completed. Status codes are broken into the following 5 blocks:

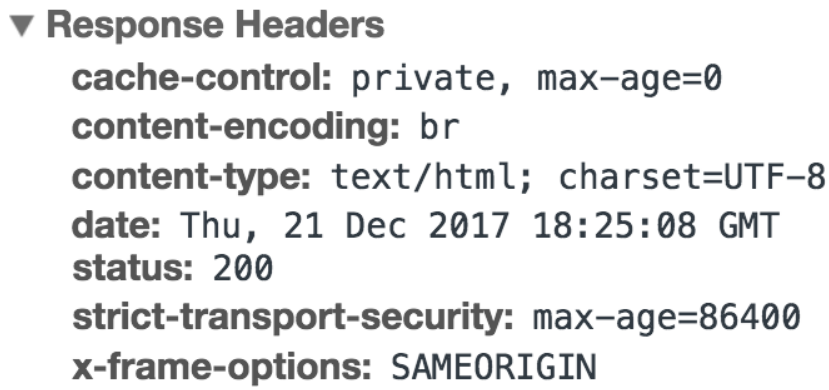
1. 1xx Informational

2. 2xx Success
3. 3xx Redirection
4. 4xx Client Error
5. 5xx Server Error

For example, a status code of 200 indicates that the request has been successfully completed. On the other hand, a status code of 404 indicates that the requested resource was not found. Find a list of all HTTP status codes here.

HTTP response headers

A response header is an HTTP header that can be used in an HTTP response and that doesn't relate to the content of the message. Response headers, like Age, Location or Server are used to give a more detailed context of the response.



Some common response headers are:

- Server - The Server header specifies the software used to handle the request.
- Content-Type - The Content-Type header specifies the media type of the body of the response.
- Content-Length - The Content-Length header specifies the length of the body of the response.
- Content-Encoding - The Content-Encoding header specifies the encoding used for the response body.
- Keep-Alive - The Keep-Alive header specifies the period of time that the connection should be kept alive.