Web Essentials

Avinash Maskey

The Internet – Introduction (1)

- The **Internet** is a global system of interconnected computer networks that use the standard Internet protocol suite (often called TCP/IP, although not all applications use TCP) to serve billions of users worldwide.
- It is a ***network of networks*** that consists of ***millions of private***, ***public***, ***academic***, ***business***, and ***government networks***, of local to global scope, that are linked by a broad array of electronic, wireless and optical networking technologies.
- The Internet carries an extensive range of information resources and services, such as the ***inter-linked hypertext documents*** of the World Wide Web (WWW) and the infrastructure to support email.

The Internet – Introduction (2)

- **Internet** is a short form of the technical term internetwork, the result of interconnecting computer networks with special gateways or routers.
- The Internet is also often referred to as the ***Net***.
- The terms Internet and World Wide Web are often used in everyday speech without much distinction.
- However, the Internet and the World Wide Web are not one and the same.
- The ***Internet*** establishes a global data communications system between computers.
- In contrast, the ***Web*** is one of the services communicated via the Internet. It is a collection of interconnected documents and other resources, linked by ***hyperlinks*** and ***URLs***.

The Internet – History

- Its history can be traced back to the 1960s when the ARPANET was created by the US Department of Defense's Advanced Research Projects Agency.
- Over time, the ARPANET expanded, and in the 1980s, it evolved into the modern Internet.
- The development of the World Wide Web in the 1990s further fueled its growth.
- Key figures in the Internet's development include ***Vint Cerf***, ***Robert Kahn***, and ***Tim Berners-Lee***.
- The Internet has had a significant impact on how we communicate and live our daily lives.

Who founded WWW?

- The **World Wide Web (WWW)** was created by ***Sir Tim Berners-Lee***, a British computer scientist, while he was working at CERN, the European Organization for Nuclear Research, in Switzerland in 1989.
- Berners-Lee's goal was to develop a system that would allow scientists at CERN to share and access information more easily.
- He invented the **H**ypertext **T**ransfer **P**rotocol **(HTTP)**, the **H**ypertext **M**arkup **L**anguage **(HTML)**, and the first web browser, which he called ***WorldWideWeb*** was founded in 1990.
- The browser was a graphical user interface (GUI) browser that allowed users to navigate through hypertext documents on the World Wide Web.
- These technologies formed the basis of the World Wide Web and revolutionized the way we access and share information over the Internet.

Internet Vs WWW

INTERNET	WWW
A global system of interconnected computer networks that use the TCP/IP protocol to link devices worldwide.	Online content that is formatted in HTML and accessed via HTTP protocol
A massive interconnection of computer networks around the world	Service provided by the internet
Uses Transmission Control Protocol / Internet Protocol (TCP/IP)	Uses Hyper Text Transfer Protocol (HTTP)

Basic Internet Protocols (1)

- **TCP/IP (Transmission Control Protocol/Internet Protocol):**
  - This is the most widely used protocol on the Internet. It is responsible for breaking data into packets, transmitting the packets across the network, and reassembling them at the receiving end.
  - **TCP** provides reliable, ordered, and error-checked delivery of data, while **IP** provides the routing and addressing functions. Together, they form the foundation of the Internet.
- **UDP (User Datagram Protocol):**
  - This is another protocol that is used for transmitting data across the Internet. Unlike TCP, it does not provide reliability or error-checking.
  - Instead, it is used for applications where speed and efficiency are more important than reliability, such as real-time video and audio streaming.

Basic Internet Protocols (2)

- **DNS (Domain Name System):**
  - This is a protocol that translates human-readable domain names (such as ***www.example.com***) into IP addresses (such as ***192.0.2.1***) that computers can understand.
  - DNS is essentially a distributed database that maps domain names to IP addresses, allowing users to access websites and other resources using easy-to-remember domain names instead of numerical IP addresses.
- **Domain Names:**
  - A domain name is a human-readable label that is used to identify a website or other Internet resource. It is composed of one or more parts separated by dots, such as ***example.com*** or ***google.co.uk***.
  - Domain names are used in **URLs** (**U**niform **R**esource **L**ocators) to specify the address of a specific resource on the Internet, such as a webpage or a file.

Basic Internet Protocols (3)

- **How does TCP/IP works?**
  - Let's say you want to send an email to a friend. When you hit the send button, your email program (such as Gmail or Outlook) divides your email into packets of data and assigns each packet a unique address called an IP address.
  - These packets are then sent over the Internet to your friend's computer, which is identified by its IP address.

Basic Internet Protocols (4)

- **Here's how TCP/IP comes into play:**
  - The ***TCP protocol*** breaks your email into packets and adds a header to each packet. The header includes information such as the source IP address, the destination IP address, and a sequence number that ensures the packets are reassembled in the correct order.
  - The packets are then sent over the Internet using the ***IP protocol***. Each packet is routed independently from one network to another, and the IP protocol ensures that they reach their destination by finding the most efficient route.
  - When the packets arrive at your friend's computer, the TCP protocol reassembles them in the correct order based on the sequence number in the header. It then sends an acknowledgement back to your computer to confirm that the packets have been received.
  - If any packets are lost or corrupted during transmission, the TCP protocol will request that they be retransmitted until all packets are successfully delivered.
- Once all packets have been received and reassembled in the correct order, your friend's computer can then display your email in their email program.

Basic Internet Protocols (5)

- **How does UDP works?**
  - Let’s take an example of a ***video streaming application***.
  - The video streaming application breaks the video into packets and assigns each packet a unique address called a ***UDP port number***.
  - The packets are then sent over the Internet using the UDP protocol. Each packet is sent independently and may take a different route to reach its destination. There is no guarantee that the packets will arrive in the correct order or that they will all be received.
  - When the packets arrive at the receiving end, the UDP protocol simply delivers them to the application without any additional processing or error checking.
  - If any packets are lost or corrupted during transmission, there is no mechanism for requesting retransmission, so the application may experience glitches or errors in the video playback.
  - Despite its limitations, UDP is useful in situations where speed and efficiency are more important than reliability. For example, it is commonly used for video and audio streaming, online gaming, and other real-time applications where a small delay or loss of data is acceptable.

Higher Level Protocols (1)

- **Higher-level protocols** are protocols that operate on top of the TCP/IP protocol stack. These protocols use TCP/IP as a transport layer and provide additional functionality to enable communication between applications. Some examples of higher-level protocols include ***HTTP***, ***FTP***, ***SMTP***, and ***Telnet***.
- **HTTP:**
  - ***HTTP (Hypertext Transfer Protocol)*** is a protocol used for transferring data over the World Wide Web. It is the foundation of data communication on the web and is used by web browsers and servers to exchange data.
- **HTTP Request and Response Messages:**
  - ***HTTP requests and response messages*** are used to communicate between web browsers and servers. When a user enters a web address (URL) in the browser, the browser sends an HTTP request to the web server asking for the requested resource. The server then responds with an HTTP response message containing the requested resource.

Higher Level Protocols (2)

- Let's take an example of a user entering a URL in a web browser and the HTTP request and response messages that follow:
  - The user enters ***"www.example.com"*** in the web browser and presses Enter.
  - The browser sends an **HTTP GET request** to the server at "www.example.com" asking for the home page of the website.
  - The server responds with an HTTP response message containing the home page HTML file and other resources (such as images and stylesheets) needed to render the page in the browser.
  - The browser receives the response message and renders the home page in the window.

Higher Level Protocols (3)

- HTTP request and response messages ***consist of the following components***:
- **HTTP Request Message:**
  - ***Request Line:*** Contains the request method (such as **GET** or **POST**), requested resource (URL), and HTTP version.
  - ***Headers:*** Additional information about the request, such as the type of browser being used, the type of data being sent, and the type of encoding being used.
  - ***Body:*** Optional data being sent with the request, such as form data.

Higher Level Protocols (4)

- **HTTP Response Message:**
  - ***Status Line:*** Contains the HTTP version, response code (such as ***200 for success*** or ***404 for not found***), and response message.
  - ***Headers:*** Additional information about the response, such as the type of data being sent, the type of encoding being used, and the date and time of the response.
  - ***Body:*** The requested resource, such as an HTML page, image, or video.

Higher Level Protocols (5) – *HTTP Request URI*

Scheme Name	Example URL	Type of Resource
ftp	ftp://ftp.example.org/pub/afile.txt	File located on FTP server
telnet	telnet://host.example.org/	Telnet server
mailto	mailto:someone@example.org	Mailbox
https	https://secure.example.org/sec.txt	Resource on web server supporting encrypted communication
file	file:///C:/temp/localFile.txt	File accessible from machine processing this URL

Table 1: Some Non-http URL Schemes

Higher Level Protocols (6) – *HTTP Versions*

Version	Year introduced	Current status
HTTP/0.9	1991	Obsolete
HTTP/1.0	1996	Obsolete
HTTP/1.1	1997	Standard
HTTP/2	2015	Standard
HTTP/3	2022	Standard

Table 2: Summary of HTTP milestone versions

Higher Level Protocols (7) – *HTTP Request Method*

- The standard HTTP methods and a brief description of each are shown in **Table 3**.
- The method part of the start line of an HTTP request must be written entirely in ***uppercase letters***, as shown in the table.
- In addition to the methods shown, the HTTP/1.1 standard defines a CONNECT method, which can be used to create certain types of secure connections.
- However, its use is beyond our scope and therefore will not be discussed further here.

Higher Level Protocols (8) – *HTTP Request Method*

- The ***primary HTTP method is GET***. This is the method used when you type a URL into the location bar of your browser.
- It is also the method that is ***used by default*** when you click on a link in a document displayed in your browser and when the browser downloads images for display within an HTML document.
- The ***POST method*** is typically used to send information collected from a form displayed within a browser, such as an order-entry form, back to the web server.
- The other methods are not frequently used by web developers, and we will therefore not discuss them further here.

Higher Level Protocols (9) – *HTTP Request Method*

Method	Requests server to . . .
GET	return the resource specified by the Request-URI as the body of a response message.
POST	pass the body of this request message on as data to be processed by the resource specified by the Request-URI.
HEAD	return the same HTTP header fields that would be returned if a GET method were used, but not return the message body that would be returned to a GET (this provides information about a resource without the communication overhead of transmitting the body of the response, which may be quite large).
OPTIONS	return (in Allow header field) a list of HTTP methods that may be used to access the resource specified by the Request-URI.
PUT	store the body of this message on the server and assign the specified Request-URI to the data stored so that future GET request messages containing this Request-URI will receive this data in their response messages.
DELETE	respond to future HTTP request messages that contain the specified Request-URI with a response indicating that there is no resource associated with this Request-URI.
TRACE	return a copy of the complete HTTP request message, including start line, header fields, and body, received by the server. Used primarily for test purposes.

Table 3: Standard HTTP/1.1 Methods

Higher Level Protocols (10) – *HTTP Request Header Fields*

Field Name	Use
Host	Specify authority portion of URL (host plus port number). Used to support virtual hosting (running separate web servers for multiple fully qualified domain names sharing a single IP address).
User-Agent	A string identifying the browser or other software that is sending the request.
Accept	MIME types of documents that are acceptable as the body of the response, possibly with indication of preference ranking. If the server can return a document according to one of several formats, it should use a format that has the highest possible preference rating in this header.
Accept-Language	Specifies preferred language(s) for the response body. A server may have several translations of a document, and among these should return the one that has the highest preference rating in this header field.
Accept-Encoding	Specifies preferred encoding(s) for the response body. For example, if a server wishes to send a compressed document (to reduce transmission time), it may only use one of the types of compression specified in this header field.

Table 4: Some Common HTTP/1.1 Request Header Fields

Higher Level Protocols (11) – *HTTP Request Header Fields*

Field Name	Use
Accept-Charset	Allows the client to express preferences to a server that can return a document using various character sets.
Connection	Indicates whether or not the client would like the TCP connection kept open after the response is sent. Typical values are keep-alive if connection should be kept open (the default behavior for servers/clients compatible with HTTP/1.1), and close if not.
Keep-Alive	Number of seconds TCP connection should be kept open.
Content-Type	The MIME type of the document contained in the message body, if one is present. If this field is present in a request message, it normally has the value shown in the example, application/x-www-form-urlencoded.

Table 4: Some Common HTTP/1.1 Request Header Fields

Higher Level Protocols (12) – *HTTP Request Header Fields*

Field Name	Use
Content-Length	Number of bytes of data in the message body, if one is present.
Referer	The URI of the resource from which the browser obtained the Request-URI value for this HTTP request. For example, if the user clicks on a hyperlink in a web page, causing an HTTP request to be sent to a server, the URI of the web page containing the hyperlink will be sent in the Referer field of the request. This field is not present if the HTTP request was generated by the user entering a URI in the browser's Location bar.

Table 4: Some Common HTTP/1.1 Request Header Fields

Higher Level Protocols (13) – *HTTP Response Header Fields*

Status Code	Recommended Reason Phrase	Usual Meaning
200	OK	Request processed normally.
301	Moved Permanently	URI for the requested resource has changed. All future requests should be made to URI contained in the Location header field of the response. Most browsers will automatically send a second request to the new URI and display the second response.
307	Temporary Redirect	URI for the requested resource has changed at least temporarily. This request should be fulfilled by making a second request to URI contained in the Location header field of the response. Most browsers will automatically send a second request to the new URI and display the second response.
401	Unauthorized	The resource is password protected, and the user has not yet supplied a valid password.
403	Forbidden	The resource is present on the server but is read protected (often an error on the part of the server administrator, but may be intentional).
404	Not Found	No resource corresponding to the given Request-URI was found at this server.
500	Internal Server Error	Server software detected an internal failure.

Table 5: Some Common HTTP/1.1 Status Codes

Higher Level Protocols (14) – *HTTP MIME Types*

- ***HTTP MIME Types***, or ***Multipurpose Internet Mail Extensions*** types, are a way of identifying files on the internet based on their content or purpose.
- MIME types are a part of the HTTP protocol, which is used to transfer data over the internet, and are used to describe the type of data that is being sent.
- MIME types are typically represented as a string of text that includes two parts: ***a type*** and ***a subtype***, separated by a forward slash. **For example,** the MIME type for HTML documents is ***"text/html"***, where ***"text"*** is the type and ***"html"*** is the subtype.
- MIME types are important because they help web servers and web browsers determine how to handle different types of files.
- **For example,** when a web browser requests a file from a web server, it will look at the MIME type of the file to determine how to display it. If the MIME type is "text/html", the browser will display the file as an HTML document. If the MIME type is "image/jpeg", the browser will display the file as a JPEG image.
- The list of registered MIME types is maintained by the **I**nternet **A**ssigned **N**umbers **A**uthority **(IANA)**.

Higher Level Protocols (15) – *HTTP MIME Types*

Top-level Content Type	Document Content
application	Data that does not fit within another content type and that is intended to be processed by application software, or that is itself an executable binary.
audio	Audio data. Subtype defines audio format.
image	Image data, typically static. Subtype defines image format. Requires appropriate software and hardware in order to be displayed.
message	Another document that represents a MIME-style message. For example, following an HTTP TRACE request message to a server, the server sends a response with a body that is a copy of the HTTP request. The value of the Content-Type header field in the response is message/http.
model	Structured data, generally numeric, representing physical or behavioral models.
multipart	Multiple entities, each with its own header and body.
text	Displayable as text. That is, a human can read this document without the need for special software, although it may be easier to read with the assistance of other software.
video	Animated images, possibly with synchronized sound.

Table 6: Standard Top-level MIME Content Types

Higher Level Protocols (16) – *HTTP MIME Types*

MIME Type	Description
text/html	HTML document
image/gif	Image represented using Graphics Interchange Format (GIF)
image/jpeg	Image represented using Joint Picture Expert Group (JPEG) format
text/plain	Human-readable text with no embedded formatting information
application/octet-stream	Arbitrary binary data (may be executable)
application/x-www-form-urlencoded	Data sent from a web form to a web server for processing

Table 7: Some Common MIME Content Types

Higher Level Protocols (17) – *HTTP Header Fields*

- The following is a modified example of an actual HTTP request sent by a browser consists of a start line, 10 header fields, and a short message body:

```
POST /servlet/EchoHttpRequest HTTP/1.1
host: www.example.org:56789
user-agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.4)
  Gecko/20030624
accept: text/xml,application/xml,application/xhtml+xml,
  text/html;q=0.9,text/plain;q=0.8,video/x-mng,image/png,image/jpeg,
  image/gif;q=0.2,*/*;q=0.1
accept-language: en-us,en;q=0.5
accept-encoding: gzip,deflate
accept-charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
connection: keep-alive
keep-alive: 300
content-type: application/x-www-form-urlencoded
content-length: 13

doit=Click+me
```

URL – Uniform Resource Locator (1)

- A **URL** or **"Uniform Resource Locator"** specifies where you can find a resource on the web; you are probably most used to think of them as web addresses. As you move around the Web, you will see the URL of each web page in the address bar of your browser.

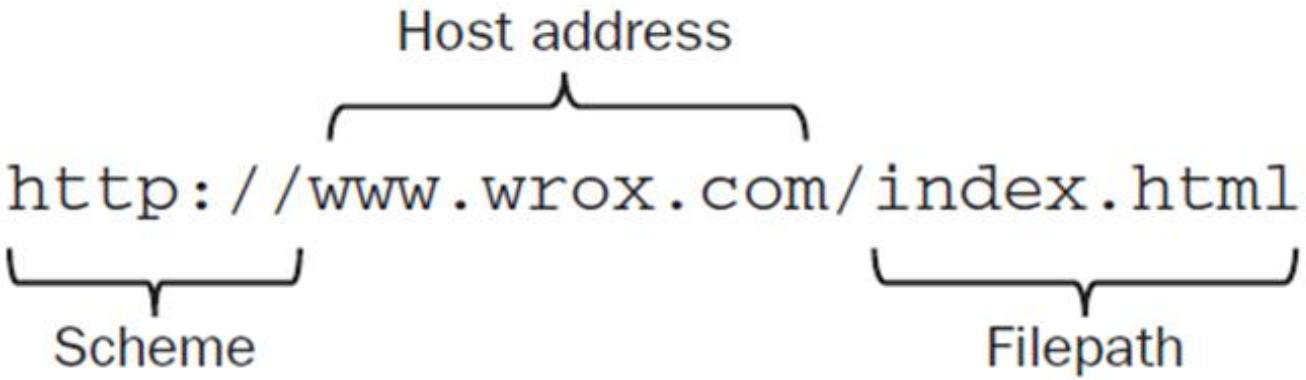


Fig: URL

URL – Uniform Resource Locator (2)

If you look at the example, there are three key parts to the URL:

- **The scheme:** Specifies how the resource is to be accessed.
- **The host address/server address:** Specifies the name of the computer where the resource is located.
- **The filepath:** Specifies the sequence of directories leading to the target. If resource is omitted, the target is the last directory in path.
- **The resource:** If included, resource is the target, and is ***typically the name of a file***. It may be a simple file, containing a single binary stream of bytes, or a structured document, containing one or more storages and binary streams of bytes.

URI Vs URL (1)

- A **URI (Uniform Resource Identifier)** is a string of characters that identifies a resource. It can be a name, a location, or both. A URI can be used to identify any type of resource, including web pages, images, files, and more.
- **Example of a URI:**

  spotify:album:2noRn2Aes5aoNVsU6iWThc
- This is a URI that identifies a specific album on the Spotify platform. It doesn't provide information on how to access the album, but it still serves as a unique identifier for the resource.

URI Vs URL (2)

- A **URL (Uniform Resource Locator)** is a type of URI that specifies the exact location of a resource on the internet. In other words, a URL is a specific type of URI that includes information on how to access a resource, such as the protocol (http, https, ftp, etc.), the domain name, the path, and any query parameters or fragments.
- **Example of a URL:**

  https://www.google.com/search?q=difference+between+uri+and+url
- This is a URL that provides the location of a specific web page on the Google search engine.
- It includes the protocol (https), the domain name (www.google.com), and the path (/search) with query parameters (?q=difference+between+uri+and+url).
- To summarize, a ***URI is a general identifier for a resource***, while a ***URL is a specific type of URI that includes information on how to access the resource***.

Client-Server Architecture (1)

- **Client-server architecture** is a common design pattern used in web technology, where the system is divided into two parts: ***the client***, which is responsible for sending requests to the server, and ***the server***, which is responsible for processing these requests and sending back the response.
- In web technology, the client is typically a web browser, and the server is typically a web server running server-side code. Here's an explanation of how client-server architecture works in web technology:
  - The user enters a URL into the web browser.
  - The web browser sends a request to the web server.
  - The web server receives the request and runs server-side code to generate a response.
  - The web server sends the response back to the web browser.
  - The web browser receives the response and renders it on the screen.

Client-Server Architecture (2)

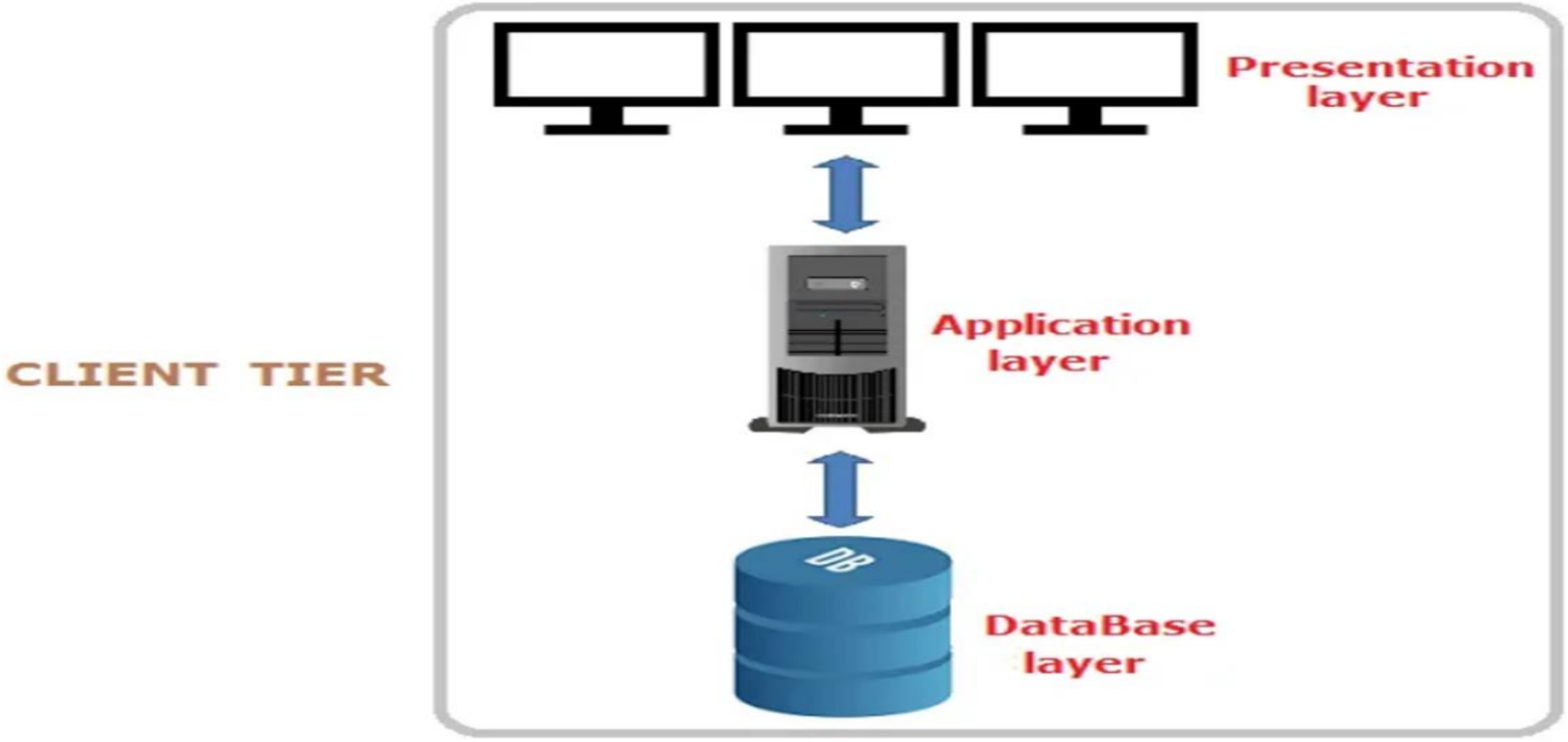
- In this architecture, the client and server communicate using a protocol called **HTTP** (Hypertext Transfer Protocol).
- When the client sends a request to the server, it includes an HTTP method (such as GET or POST) and a URL (Uniform Resource Locator) that identifies the resource the client is requesting.
- The server processes the request and generates an HTTP response, which includes a status code (such as 200 for success or 404 for not found) and the content of the response (such as HTML, CSS, or JavaScript).

Client-Server Architecture (3)

- Client-server architecture in web technology is scalable and flexible, as the server can handle multiple requests from different clients simultaneously, and the client can be running on any device with a web browser, including desktops, laptops, tablets, and smartphones.
- This architecture is used for a wide variety of web applications, including ***e-commerce websites***, ***social media platforms***, and ***online banking systems***.

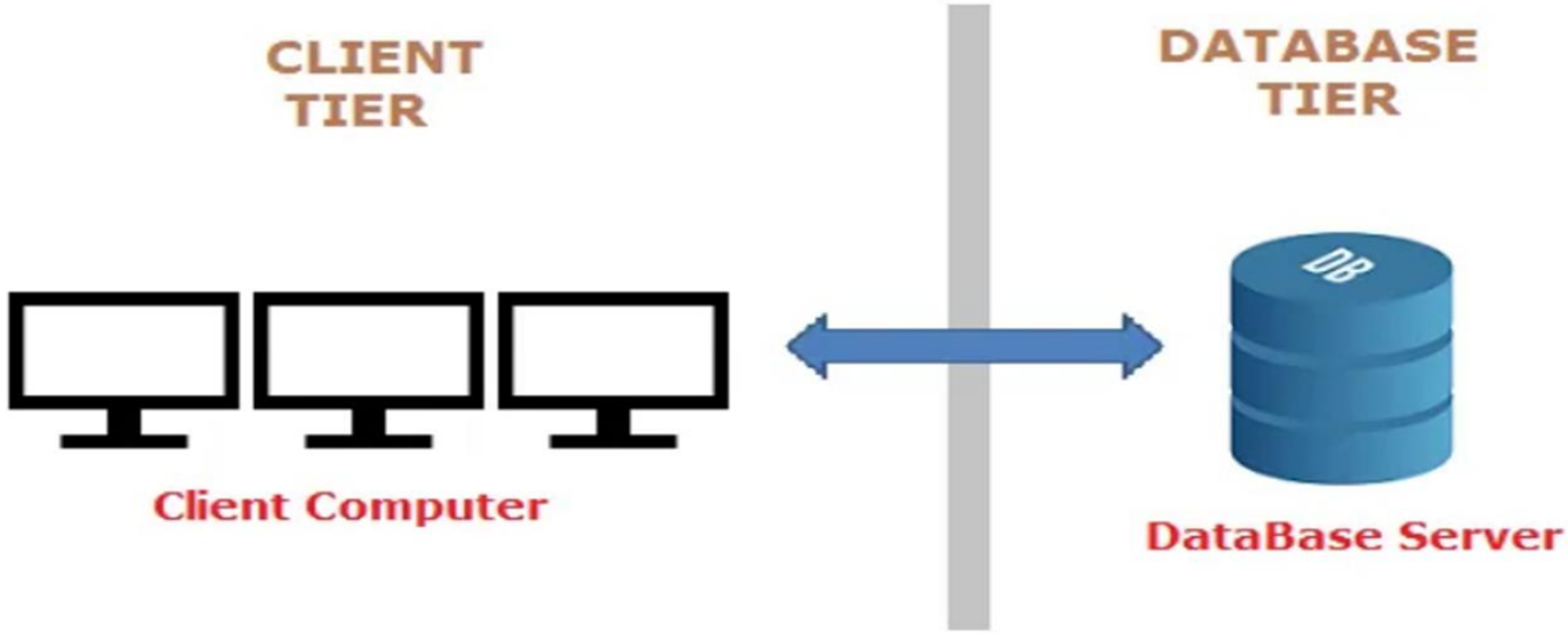
Client-Server Architecture (4) – *One Tier Architecture*

- In a **one-tier architecture**, also known as a ***monolithic architecture***, the entire application runs on a single machine. The ***user interface, application logic, and database are all on the same machine, and the client and server are tightly coupled***. This architecture is simple and easy to implement but can be difficult to scale.



Client-Server Architecture (5) – *Two Tier Architecture*

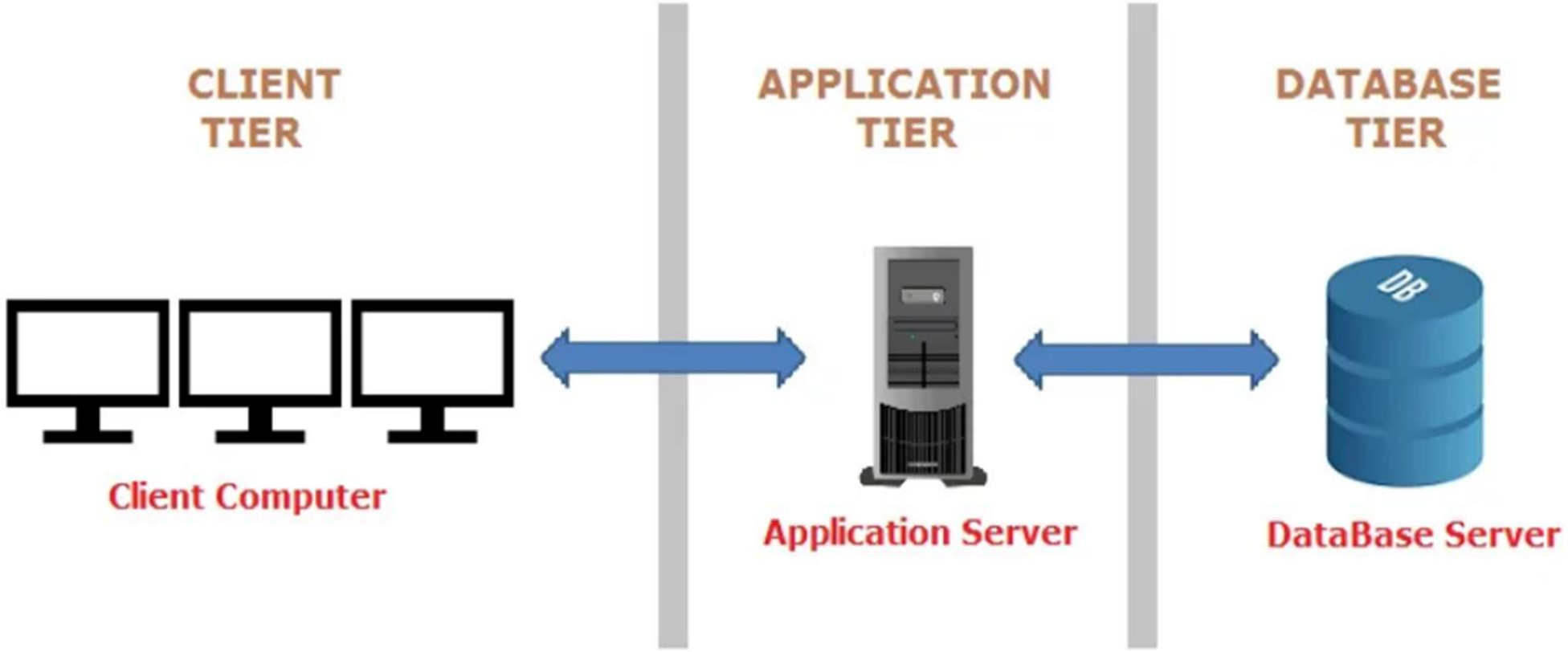
- In a **two-tier architecture**, also known as a ***client-server architecture***, the application is split into two parts: ***a client*** and ***a server***. The client is responsible for the user interface and runs on the user's machine, while the server is responsible for the application logic and database and runs on a remote server. This architecture is more scalable than a one-tier architecture, as the server can handle multiple clients simultaneously.



Client-Server Architecture (6) – *Three Tier Architecture*

- In a **three-tier architecture**, the application is split into three parts: ***a client***, ***an application server***, and ***a database server***.
- The client is responsible for the user interface, the application server is responsible for the application logic, and the database server is responsible for the data storage.
- This architecture is even more scalable than a two-tier architecture, as the application server can handle multiple clients simultaneously and the database server can handle large amounts of data.
- In web technology, ***the client is typically a web browser, the application server is a web server running server-side code (e.g. PHP, Python, Ruby)***, ***and the database server is a database management system (e.g. MySQL, PostgreSQL, MongoDB)***.

Client-Server Architecture (6) – *Three Tier Architecture*



Web 1.0 Vs Web 2.0 Vs Web 3.0 (1)

Web 1.0	Web 2.0	Web 3.0
From early 1990s to mid-1990s	From early 2000s to mid-2000s	From late 2000s to present day
Mostly Read-Only	Wildly Read-Write	Portable and Personal
Company Focus	Community Focus	Individual Focus
Home Pages	Blogs / Wikis	Live-streams / Waves
Owning Content	Sharing Content	Consolidating Content
WebForms	Web Applications	Smart Applications
Directories	Tagging	User behavior
Page Views	Cost Per Click	User Engagement
Banner Advertising	Interactive Advertising	Behavioral Advertising
Britannica Online	Wikipedia	The Semantic Web
HTML/Portals	XML / RSS	RDF / RDFS / OWL
Data was not Focused.	Data of many was controlled by some mediatory.	Data was personalized and no use of mediatory.

Web 1.0 Vs Web 2.0 Vs Web 3.0 (2)

Web 1.0	Web 2.0	Web 3.0
Information sharing is the goal.	Interaction is the goal.	Immersion is the goal.
It connects information as its primary goal.	It aims to connect people.	Focuses on relating knowledge.
Static websites	Introduction of web applications	Intelligent web-based functions and apps
A simpler, more passive web.	An enhanced social Web	A semantic web exists.
Web and File Servers, HTML, and Portals are technologies connected to Web 1.0.	AJAX, JavaScript, CSS, and HTML5 are examples of related technology.	Web 3.0 technologies include blockchain, artificial intelligence, and decentralized protocols.
Associated Technologies • Web and File Servers • Search Engines (including AltaVista and Yahoo!) • E-mail accounts (Yahoo!, Hotmail) • Peer-to-Peer File Sharing (Napster, BitTorrent) and others.	**Associated Technologies** • Frameworks for Ajax and JavaScript • Microsoft.NET • Blogs • Wikis and others.	**Associated Technologies** • Searching Using Semantics • Databases of Information • Ontologies • Intelligent Digital Personal Assistants and others.

THANK YOU!