History of the Internet

The first recorded description of the social interactions that could be enabled through networking was a series of memos written by J.C.R. Licklider of MIT in August 1962 discussing his "Galactic Network" concept. He envisioned a globally interconnected set of computers through which everyone could quickly access data and programs from any site. In spirit, the concept was very much like the Internet of today. Licklider was the first head of the computer research program at DARPA,[4] starting in October 1962. While at DARPA he convinced his successors at DARPA, Ivan Sutherland, Bob Taylor, and MIT researcher Lawrence G. Roberts, of the importance of this networking concept.

In late 1966 Roberts went to DARPA to develop the computer network concept and quickly put together his plan for the "ARPANET", publishing it in 1967. At the conference where he presented the paper, there was also a paper on a packet network concept from the UK by Donald Davies and Roger Scantlebury of NPL. Scantlebury told Roberts about the NPL work as well as that of Paul Baran and others at RAND. The RAND group had written a paper on packet switching networks for secure voice in the military in 1964. It happened that the work at MIT (1961-1967), at RAND (1962-1965), and at NPL (1964-1967) had all proceeded in parallel without any of the researchers knowing about the other work. The word "packet" was adopted from the work at NPL and the

proposed line speed to be used in the ARPANET design was upgraded from 2.4 kbps to 50 kbps.[5]

In August 1968, after Roberts and the **DARPA** funded community had refined the overall structure and specifications for the ARPANET.

Computers were added quickly to the **ARPANET** during the following years, and work proceeded on completing a functionally complete Host-to-Host protocol and other network software. In December 1970 the Network Working Group (NWG) working under S. Crocker finished the initial ARPANET Host-to-Host protocol, called the Network Control Protocol (NCP). As the ARPANET sites completed implementing NCP during the period 1971-1972, the network users finally could begin to develop applications.

The idea of **open-architecture networking** was first introduced by Kahn shortly after having arrived at DARPA in 1972.

The original ARPANET grew into the Internet. Internet was based on the idea that there would be multiple independent networks of rather arbitrary design, beginning with the ARPANET as the pioneering packet switching network,

but soon to include packet satellite networks, ground-based packet radio networks and other networks

In an open-architecture network, the individual networks may be separately designed and developed and each may have its own unique
interface which it may offer to users and/or other providers. including other Internet providers. Each network can be designed in accordance with the specific environment and user requirements of that network. There are generally no constraints on the types of network that can be included or on their geographic scope, although certain pragmatic considerations will dictate what makes sense to offer.

The original Cerf/Kahn paper on the Internet described one protocol, called **TCP**, which provided all the transport and forwarding services in the Internet. Kahn had intended that the TCP protocol support a range of transport services, from the totally reliable sequenced delivery of data (virtual circuit model) to a datagram service in which the application made direct use of the underlying network service, which might imply occasional lost, corrupted or reordered packets.

DARPA let three contracts to Stanford (Cerf), BBN (Ray Tomlinson) and UCL (Peter Kirstein) to implement **TCP/IP** (it was simply called TCP in the Cerf/ Kahn paper but contained both components). The Stanford team, led by Cerf,

produced the detailed specification and within about a year there were three independent implementations of TCP that could interoperate.

Widespread development of LANS, PCs and workstations in the 1980s
allowed the nascent Internet to flourish. Ethernet technology, developed by Bob Metcalfe at Xerox PARC in 1973, is now probably the dominant network technology in the Internet and PCs and workstations the dominant computers. A major shift occurred as a result of the increase in scale of the Internet and its associated management issues

Originally, there were a fairly limited number of hosts, so it was feasible to maintain a single table of all the hosts and their associated names and addresses. The shift to having a large number of independently managed networks (e.g., LANs) meant that having a single table of hosts was no longer feasible, and the **Domain Name System (DNS)** was invented by Paul Mockapetris of USC/ISI. The DNS permitted a scalable distributed mechanism for resolving hierarchical host names (e.g. www.acm.org) into an Internet address.

One of the more interesting challenges was the transition of the ARPANET host protocol from **NCP to TCP/IP** as of January 1, 1983.

By 1983, ARPANET was being used by a significant number

of defense R&D and operational organizations. The transition of ARPANET from NCP to TCP/IP permitted it to be split into a MILNET supporting operational requirements and an ARPANET supporting research needs.

Thus, by 1985, Internet was already well established as a technology supporting a broad community of researchers and developers, and was beginning to be used by other communities for daily computer communications. Electronic mail was being used broadly across several communities, often with different systems, but interconnection between different mail systems was demonstrating the utility of broad based electronic communications between people.

In 1985, Dennis Jennings came from Ireland to spend a year at NSF leading the **NSFNET** program. He worked with the community to help NSF make a critical decision - that TCP/IP would be mandatory for the NSFNET program. When Steve Wolff took over the NSFNET program in 1986, he recognized the need for a wide area networking infrastructureto support the general academic and research community, along with the need to develop a strategy for establishing such infrastructure on a basis ultimately independent of direct federal funding. Policies and strategies were adopted (see below) to achieve that end.

. Such was the weight of the NSFNET program's

ecumenism and funding ($200 million from 1986 to 1995) - and the quality of the protocols themselves - that by 1990 when the ARPANET itself was finally decommissioned[10], TCP/IP had supplanted or marginalized most other wide-area computer network protocols worldwide, and IP was well on its way to becoming THE bearer service for the Global Information Infrastructure.

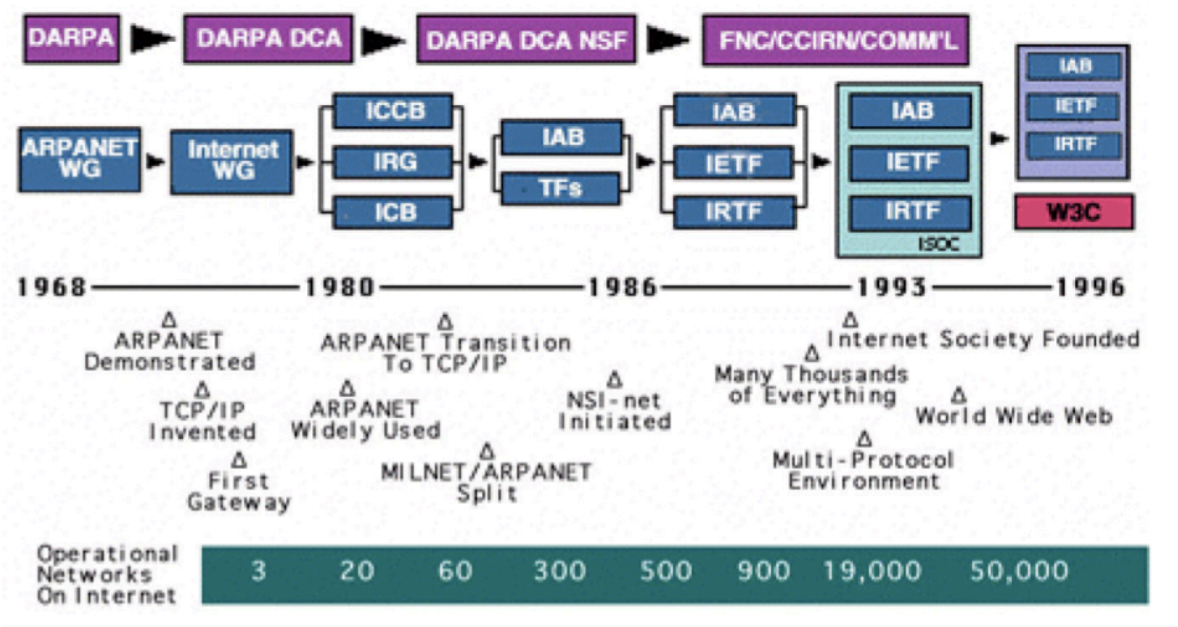
Formation of the Broad Community

The Internet is as much a collection of communities as a collection of technologies, and its success is largely attributable to both satisfying basic community needs as well as utilizing the community in an effective way to push the infrastructure forward. This community spirit has a long history beginning with the early ARPANET.

The recent development and widespread deployment of the World Wide Web has brought with it a new community, as many of the people working on the WWW have not thought of themselves as primarily network researchers and developers. A new coordination organization was formed, the **World Wide Web Consortium (W3C).** Initially led from MIT's Laboratory for Computer Science by **Tim Berners-Lee** (the inventor of the **WWW**) , W3C has taken on the responsibility for evolving the various protocols and standards associated with the Web.

Commercialization of the Technology

After two years of conferences, tutorials, design meetings and workshops, a special event was organized that invited those vendors whose products ran TCP/IP well enough to come together
in one room for three days to show off how well they all worked together and also ran over the Internet. In September of 1988 the first Interop trade show was born. 50 companies made the cut. 5,000 engineers from potential customer organizations came to see if it all did work as was promised. It did. Why? Because the vendors worked extremely hard to ensure that everyone's products interoperated with all of the other products - even with those of their competitors. The Interop trade show has grown immensely since then and today it is held in 7 locations around the world each year to an audience of over 250,000 people who come to learn which products work with each other in a seamless manner, learn about the latest products, and discuss the latest technology.



Summary Video

https://www.youtube.com/watch?v=n-Cd6uQSiGA

How Does The Internet Work?

The Internet works through a packet routing network in accordance with the Internet Protocol (IP), the Transport Control Protocol (TCP) and other protocols.

What's a protocol?

A protocol is a set of rules specifying how computers should communicate with each other over a network. For example, the Transport Control Protocol has a rule that if one computer sends data to another computer, the destination computer should let the source computer know if any data was missing so the source computer can re-send it. Or the Internet Protocol

which specifies how computers should route information to other computers by attaching addresses onto the data it sends.

What's a packet?

Data sent across the Internet is called a message. Before a message is sent, it is first split in many fragments called packets. These packets are sent independently of each other. The typical maximum packet size is between 1000 and 3000 characters. The Internet Protocol specifies how messages should be packetized.

What's a packet routing network?

It is a network that routes packets from a source computer to a destination computer. The Internet is made up of a massive network of specialized computers called routers. Each router's job is to know how to move packets along from their source to their destination. A packet will have moved through multiple routers during its journey.

When a packet moves from one router to the next, it's called a hop. You can use the command line-tool traceroute to see the list of hops packets take between you and a host.

The Internet Protocol specifies how network addresses should be attached to the packet's headers, a designated space in the

packet containing its meta-data. The Internet Protocol also specifies how the routers should forward the packets based on the address in the header.

Do the packets always arrive in order? If not, how is the message re- assembled?

The packets may arrive at their destination out of order. This happens when a later packet finds a quicker path to the destination than an earlier one. But packet's header contains information about the packet's order relative to the entire message. The Transport Control Protocol uses this info for reconstructing the message at the destination.

Do packets always make it to their destination?

The Internet Protocol makes no guarantee that packets will always arrive at their destinations. When that happens, it's called called a packet loss. This typically happens when a router receives more packets it can process. It has no option other than to drop some packets.

However, **the Transport Control Protocol handles packet loss by performing re- transmissions**. It does this by having the destination computer periodically send

acknowledgement packets back to the source computer

indicating how much of the message it has received and reconstructed. If the destination computer finds there are missing packets, it sends a request to the source computer asking it to resend the missing packets.

When two computers are communicating through the Transport Control Protocol, we say there is a TCP connection between them.

What do these Internet addresses look like?

These addresses are called IP addresses and there are two standards.

The first address standard is called IPv4 and it looks like 212.78.1.25 . But because IPv4 supports only 232 (about 4 billion) possible addresses, the Internet Task Force proposed a new address standard called IPv6, which look like

3ffe:1893:3452:4:345:f345:f345:42fc .

IPv6 supports 2128 possible addresses, allowing for much more networked devices, which will be plenty more than the as of 2017 current 8+ billion networked devices on the Internet.

As such, there is a one-to-one mapping between IPv4 and IPv6 addresses. Note the switch from IPv4 to IPv6 is still in progress and will take a long time. As of 2014, Google revealed their IPv6 traffic was only at 3%.

How can there be over 8 billion networked devices on the Internet if there are only about 4 billion IPv4 addresses?

It's because there are public and private IP addresses. Multiple devices on a local network connected to the Internet will share the same public IP address. Within the local network, these devices are differentiated from each other by private IP addresses, typically of the form 192.168.xx or 172.16.x.x or 10.x.x.x where x is a number between 1 and 255. These private IP addresses are assigned by Dynamic Host Configuration Protocol (DHCP).

For example, if a laptop and a smart phone on the same local network both make a request to www.google.com, before the packets leave the modem, it modifies the packet headers and assigns one of its ports to that packet. When the google server responds to the requests, it sends data back to the modem at this specific port, so the modem will know whether to route the packets to the laptop or the smart phone.

In this sense, IP addresses aren't specific to a computer, but more the connection which the computer connects to the Internet with. The address that is unique to your computer is the MAC address, which never changes throughout the life of the computer.

This protocol of mapping private IP addresses to public IP addresses is called the Network Address Translation (NAT) protocol. It's what makes it possible to support 8+ billion networked devices with only 4 billion possible IPv4

addresses.

How does the router know where to send a packet?

Every router does not need to know where every IP address is. It only needs to know which one of its neighbors, called an outbound link, to route each packet to. Note that IP Addresses can be broken down into two parts, a network prefix and a host identifier. For example, 129.42.13.69 can be broken down into

Network Prefix: 129.42 Host Identifier: 13.69

All networked devices that connect to the Internet through a single connection (ie. college campus, a business, or ISP in metro area) will all share the same network prefix.

Routers will send all packets of the form 129.42.*.* to the same location. So instead of keeping track of billions of IP addresses, routers only need to keep track of less than a million network prefix.

But a router still needs to know a lot of network prefixes .

If a new router is added to the Internet how does it know how to handle packets for all these network prefixes?

A new router may come with a few preconfigured routes. But if it encounters a packet it does not know how to route, it queries one of its neighbouring routers. If the neighbour

knows how to route the packet, it sends that info back to the requesting router. The requesting router will save this info for future use. In this way, a new router builds up its own routing table, a database of network prefixes to outbound links. If the neighbouring router does not know, it queries its neighbours and so on.

How do networked computers figure out ip addresses based on domain names?

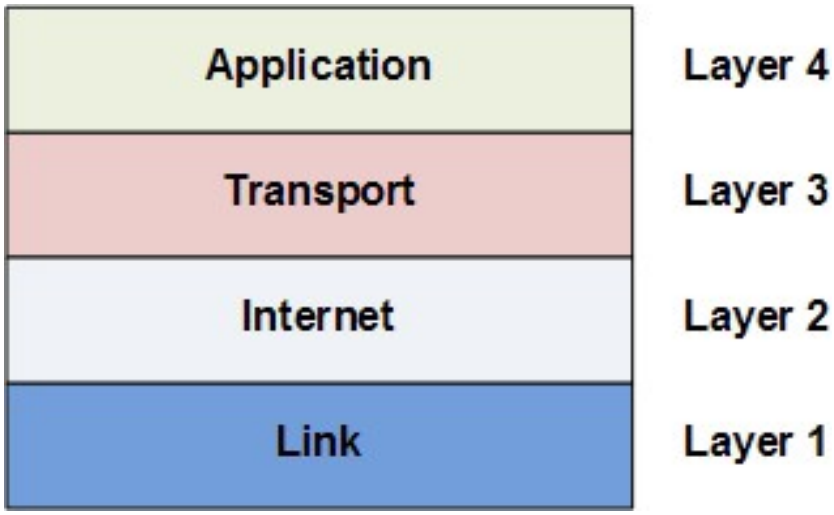
We call looking up the IP address of a human-readable domain name like www.google.com "resolving the IP address". Computers resolve IP addresses through the

Domain Name System (DNS), a decentralized database of mappings from domain names to IP addresses.

To resolve an IP address, the computer first checks its *local DNS cache*, which stores the IP address of web sites it has visited recently. If it can't find the IP address there or that IP address record has expired, it queries the ISP's DNS servers which are dedicated to resolving IP addresses. If the ISP's DNS servers can't find resolve the IP address, they query the root name servers, which can resolve every domain name for a given top-level domain . Top-level domains are the words to the right of the right-most period in a domain name. .com .net .org are some examples of top-level domains.

How do applications communicate over the Internet?

Like many other complex engineering projects, the Internet is broken down into smaller independent components, which work together through well-defined interfaces. These components are called the Internet Network Layers and they consist of Link Layer, Internet Layer, Transport Layer, and Application Layer. These are called layers because they are built on top of each other; each layer uses the capabilities of the layers beneath it without worrying about its implementation details.



Internet applications work at the Application Layer and don't need to worry about the details in the underlying layers. For example, an application connects to another application on the network via TCP using a construct called a socket, which

abstracts away the gritty details of routing packets and re-assembling packets into messages.

What do each of these Internet layers do?

At the lowest level is the Link Layer which is the “physical layer” of the Internet. The Link Layer is concerned with transmitting data bits through some physical medium like fiber-optic cables or wifi radio signals.

On top of the Link Layer is the Internet Layer. The Internet Layer is concerned with routing packets to their destinations. The Internet Protocol mentioned earlier lives in this layer (hence the same name). The Internet Protocol dynamically adjusts and reroutes packets based on network load or outages. Note it does not guarantee packets always make it to their destination, it just tries the best it can.

The Application Layer sits on top. This layer uses all the layers below to handle the complex details of moving the packets across the Internet. It lets applications easily make connections with other applications on the Internet with simple abstractions like sockets. The HTTP protocol which specifies how web browsers and web servers should interact lives in the Application Layer. The IMAP protocol which specifies how email clients should retrieve email lives in the

Application Layer. The FTP protocol which specifies a file-transferring protocol between file-downloading clients and file-hosting servers lives in the Application Layer.

What's a client versus a server?

While clients and servers are both applications that communicate over the Internet, clients are "closer to the user" in that they are more user-facing applications like web browsers, email clients, or smart phone apps. Servers are applications running on a remote computer which the client communicates over the Internet when it needs to.

A more formal definition is that the application that initiates a TCP connection is the client, while the application that receives the TCP connection is the server.

How can sensitive data like credit cards be transmitted securely over the Internet?

In the early days of the Internet, it was enough to ensure that the network routers and links are in physically secure locations. But as the Internet grew in size, more routers meant more points of vulnerability. Furthermore, with the advent of wireless technologies like WiFi, hackers could intercept packets in the air; it was not enough to just ensure the network hardware was physically safe. The solution to this was encryption and authentication through SSL/TLS.

What is SSL/TLS?

SSL stands for Secured Sockets Layer. TLS stands for Transport Layer Security. SSL was first developed by Netscape in 1994 but a later more secure version was devised and renamed TLS. We will refer to them together as SSL/TLS.

SSL/TLS is an optional layer that sits between the Transport Layer and the Application Layer. It allows secure Internet communication of sensitive information through encryption and authentication.

Encryption means the client can request that the TCP connection to the server be encrypted. This means all messages sent between client and server will be encrypted before breaking it into packets. If hackers intercept these packets, they would not be able to reconstruct the original message.

Authentication means the client can trust that the server is who it claims to be. This protects against man-in-the-middle attacks, which is when a malicious party intercepts the connection between client and server to eavesdrop and tamper with their communication.

We see SSL in action whenever we visit SSL-enabled

websites on modern browsers. When the browser requests a web site using the https protocol instead of http , it's telling the web server it wants an SSL encrypted connection. If the web server supports SSL, a secure encrypted connection is made and we would see a lock icon next to the address bar on the browser.

How does SSL authenticate the identity of a server and encrypt their communication?

It uses asymmetric encryption and SSL certificates.

Asymmetric encryption is an encryption scheme which uses a public key and a private key. These keys are basically just numbers derived from large primes. The private key is

used to decrypt data and sign documents. The public key is used to encrypt data and verify signed documents. Unlike symmetric encryption, asymmetric encryption means the ability to encrypt does not automatically confer the ability to decrypt. It does this by using principles in a mathematical branch called number theory.

An SSL certificate is a digital document that consists of a public key assigned to a web server. These SSL certificates are issued to the server by certificate authorities. Operating systems, mobile devices, and browsers come with a database of some certificate authorities so it can verify SSL certificates.

When a client requests an SSL-encrypted connection with a

server, the server sends back its SSL certificate. The client checks that the SSL certificate

is issued to this server
is signed by a trusted certificate authority has not expired.

The client then uses the SSL certificate's public key to encrypt a randomly generated temporary secret key and send it back to the server. Because the server has the corresponding private key, it can decrypt the client's temporary secret key. Now both client and server know this temporary secret key, so they can both use it to symmetrically encrypt the messages they send to each other. They will discard this temporary secret key after their session is over.

What happens if a hacker intercepts an SSL-encrypted session?

Suppose a hacker intercepted every message sent between the client and the server. The hacker sees the SSL certificate the server sends as well as the client's encrypted temporary secret key. But because the hacker doesn't have the private key it can't decrypt the temporarily secret key. And because it doesn't have the temporary secret key, it can't decrypt any of the messages between the client and server.

Reference Videos -

https://www.youtube.com/watch?v=9hIQjrMHTv4
https://www.youtube.com/watch?v=7_LPdttKXPc

Hypertext Transfer Protocol

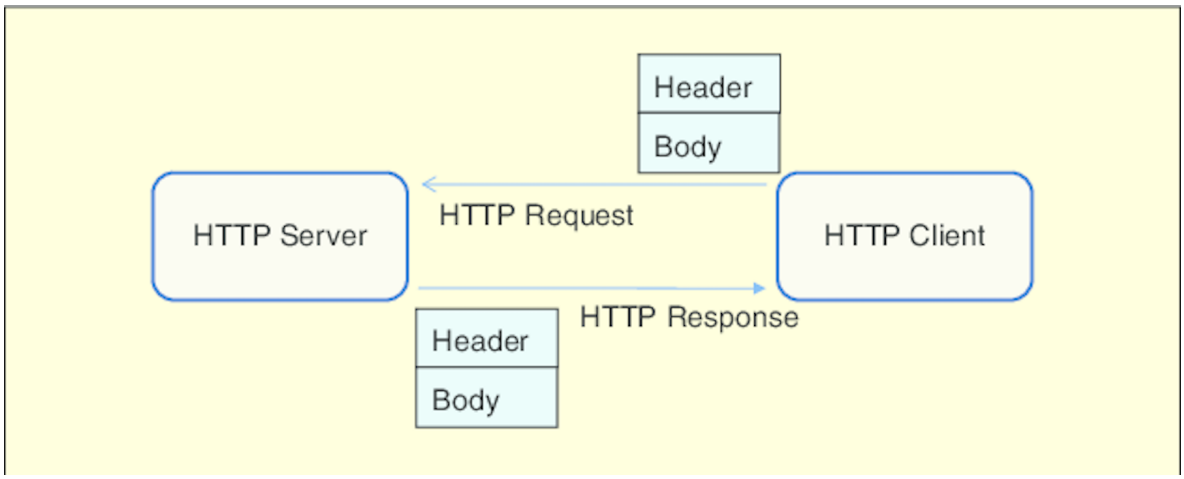
The Hypertext Transfer Protocol (HTTP) is an application protocol for distributed, collaborative, hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web, where hypertext documents include hyperlinks to other resources that the user can easily access, for example by a mouse click or by tapping the screen in a web browser.

Development of HTTP was initiated by Tim Berners-Lee at CERN in 1989. Development of early HTTP Requests for Comments (RFCs) was a coordinated effort by the Internet Engineering Task Force (IETF) and the World Wide Web Consortium (W3C), with work later moving to the IETF.

HTTP/1.1 was first documented in RFC 2068 in 1997. That specification was obsoleted by RFC 2616 in 1999, which was likewise replaced by the RFC 7230 family of RFCs in 2014.

HTTP/2 is a more efficient expression of HTTP's semantics "on the wire", and was published in 2015; it is now supported by major web servers and browsers over Transport Layer Security (TLS) using an Application-Layer Protocol Negotiation (ALPN) extension where TLS 1.2 or newer is required.

HTTP/3 is the proposed successor to HTTP/2, which is already in use on the web, using UDP instead of TCP for the underlying transport protocol. Like HTTP/2, it does not obsolete previous major versions of the protocol. Support for HTTP/3 was added to Cloudflare and Google Chrome (Canary build) in September 2019. Support in Firefox Nightly arrived in November 2019.



HTTP functions as a request–response protocol in the client–server computing model. A web browser, for example, may be the client and an application running on a computer hosting a website may be the server. The client submits an HTTP request message to the server. The server, which provides resources such as HTML files and other content, or performs other functions on behalf of the client, returns a response message to the client. The response contains completion status information about the request and may also contain requested content in its message body.

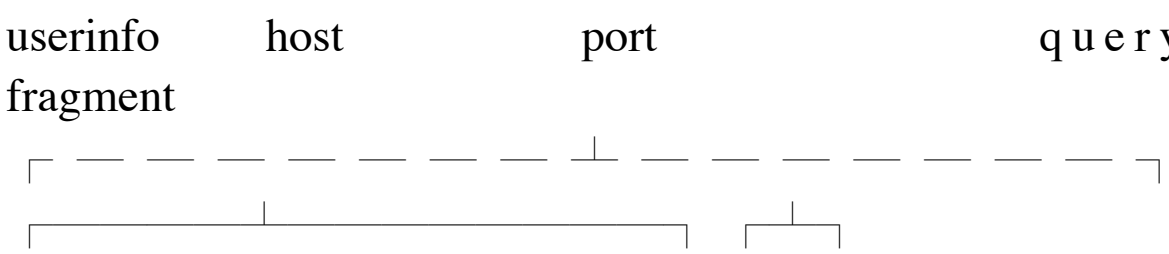
A web browser is an example of a user agent (UA). Other types of user agent include the indexing software used by search providers (web crawlers), voice browsers, mobile apps, and other software that accesses, consumes, or displays web content.

HTTP is designed to permit intermediate network elements to improve or enable communications between clients and servers. High-traffic websites often benefit from web cache servers that deliver content on behalf of upstream servers to improve response time. Web browsers cache previously accessed web resources and reuse them, when possible, to reduce network traffic. HTTP proxy servers at private network boundaries can facilitate communication for clients without a globally routable address, by relaying messages with external servers.

HTTP is an application layer protocol designed within the framework of the Internet protocol suite. Its definition presumes an underlying and reliable transport layer protocol, and Transmission Control Protocol (TCP) is commonly used. However, HTTP can be adapted to use unreliable protocols such as the User Datagram Protocol (UDP), for example in HTTPU and Simple Service Discovery Protocol (SSDP).

HTTP resources are identified and located on the network by Uniform Resource Locators (URLs), using the Uniform Resource Identifiers (URI's) schemes http and https. For example, including all optional components

URL beginning with the HTTP scheme and the WWW domain name label



As defined in RFC 3986, URIs are encoded as hyperlinks in HTML documents, so as to form interlinked hypertext documents.

Tim Berners-Lee and his team at CERN are credited with inventing the original HTTP, along with HTML and the associated technology for a web server and a text-based web browser. Berners-Lee first proposed the "WorldWideWeb" project in 1989 —now known as the World Wide Web.

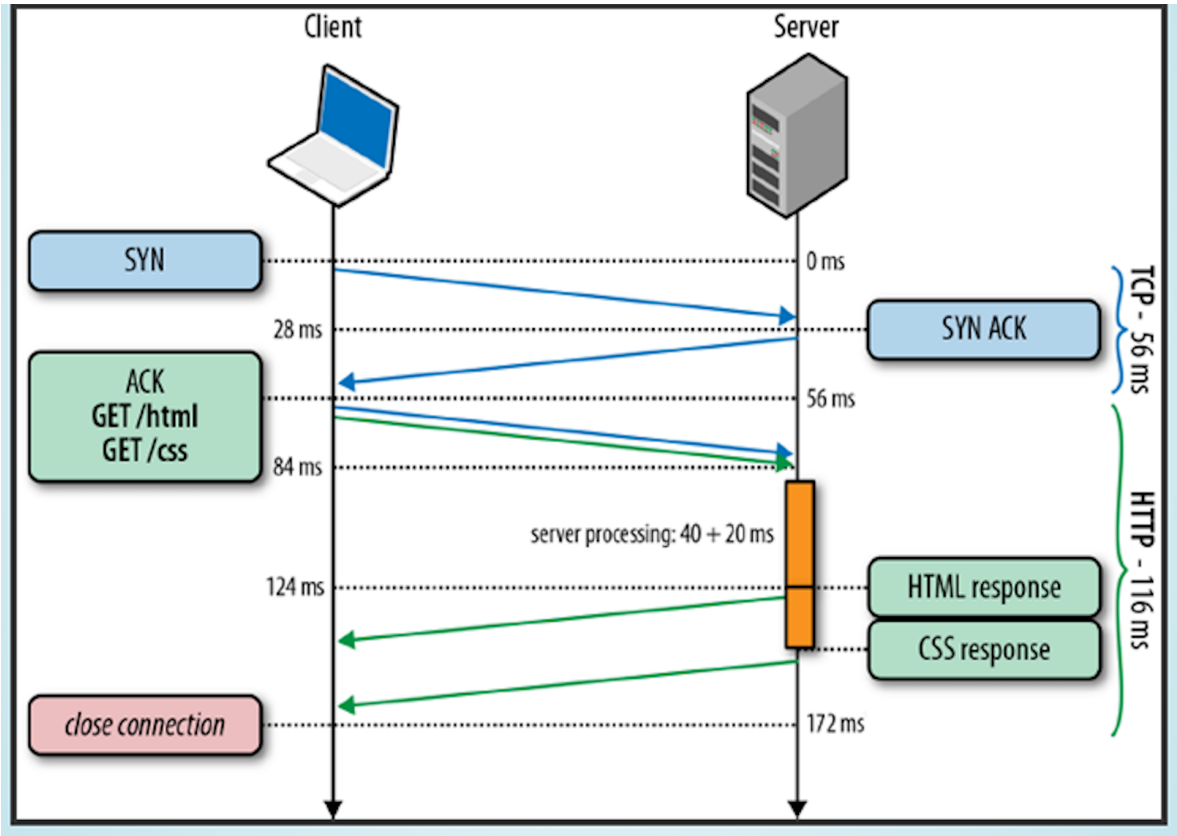
HTTP session

HTTP is a stateless protocol. A stateless protocol does not require the HTTP server to retain information or status about each user for the duration of multiple requests. However, some web applications implement states or server side sessions using for instance HTTP cookies or hidden variables within web forms.

An HTTP session is a sequence of network request-response transactions. An HTTP client initiates a request by establishing a Transmission Control Protocol (TCP) connection to a particular port on a server (typically port 80, occasionally port 8080; An HTTP server listening on that port waits for a client's request message. Upon receiving the request, the server sends back a status line, such as "HTTP/1.1 200 OK", and a message of its own. The body of this message is typically the requested resource, although an error message or other information may also be returned.

HTTP provides multiple authentication schemes such as basic access authentication and digest access authentication which operate via a challenge-response mechanism whereby the server identifies and issues a challenge before serving the requested content.

HTTP provides a general framework for access control and authentication, via an extensible set of challenge-response authentication schemes, which can be used by a server to challenge a client request and by a client to provide authentication information



Message format

- Request message

The request message consists of the following:

- a request line (e.g., GET /images/logo.png HTTP/1.1, which requests a resource called /images/logo.png from the server.)
- request header fields (e.g., Accept-Language: en).
- an empty line
- an optional message body

The request line and other header fields must each end with <CR><LF> (that is, a carriage return character followed by a line feed character). The empty line must consist of only <CR><LF> and no other whitespace.

In the HTTP/1.1 protocol, all header fields except Host are

optional.

A request line containing only the path name is accepted by servers to maintain compatibility with HTTP clients before the HTTP/1.0 specification in RFC 1945.

Request methods

HTTP defines methods (sometimes referred to as verbs, but nowhere in the specification does it mention verb, nor is OPTIONS or HEAD a verb) to indicate the desired action to be performed on the identified resource. What this resource represents, whether pre-existing data or data that is generated dynamically, depends on the implementation of the server. Often, the resource corresponds to a file or the output of an executable residing on the server. The HTTP/1.0 specification defined the **GET, HEAD and POST** methods and the HTTP/1.1 specification added five new methods: **OPTIONS, PUT, DELETE, TRACE and CONNECT**.

GET

The GET method requests a representation of the specified resource. Requests using GET should only retrieve data and should have no other effect. (This is also true of some other HTTP methods.) The W3C has published guidance principles on this distinction, saying, "Web application design should be informed by the above principles, but also by the relevant limitations.

HEAD
The HEAD method asks for a response identical to that of a GET request, but without the response body. This is useful for retrieving meta-information written in response headers, without having to transport the entire content.

POST
The POST method requests that the server accept the entity enclosed in the request as a new subordinate of the web resource identified by the URI. The data POSTed might be, for example, an annotation for existing resources; a message for a bulletin board, newsgroup, mailing list, or comment thread; a block of data that is the result of submitting a web form to a data-handling process; or an item to add to a database.

PUT
The PUT method requests that the enclosed entity be stored under the supplied URI. If the URI refers to an already existing resource, it is modified; if the URI does not point to an existing resource, then the server can create the resource with that URI.

DELETE
The DELETE method deletes the specified resource.

TRACE
The TRACE method echoes the received request so that a client can see what (if any) changes or additions have been made by intermediate servers.

OPTIONS
The OPTIONS method returns the HTTP methods that the server supports for the specified URL. This can be used to check the functionality of a web server by requesting '*' instead of a specific resource.

CONNECT
The CONNECT method converts the request connection to a transparent TCP/IP tunnel, usually to facilitate SSL-encrypted communication (HTTPS) through an unencrypted HTTP proxy.

PATCH
The PATCH method applies partial modifications to a resource.

All general-purpose HTTP servers are required to implement at least the GET and HEAD methods, and all other methods are considered optional by the specification.

Response Status codes
The first line of the HTTP response is called the status line and includes a numeric status code (such as "404") and a textual reason like "Not Found"

HTTP status code is primarily divided into five groups for better explanation of request and responses between client and server as named:

- Informational 1XX

- Successful 2XX
- Redirection 3XX
- Client Error 4XX
- Server Error 5XX

Example Status Code

100 Continue -The server, has received the request headers and the client should proceed to send the request body

200 OK - Standard response for successful HTTP requests. The actual response will depend on the request method used. In a GET request, the response will contain an entity corresponding to the requested resource. In a POST request, the response will contain an entity describing or containing the result of the action.

201 Created -The request has been fulfilled, resulting in the creation of a new resource.

202 Accepted - The request has been accepted for processing, but the processing has not been completed. The request might or might not be eventually acted upon, and may be disallowed when processing occurs

305 Use Proxy (since HTTP/1.1) - The requested resource is available only through a proxy, the address for which is provided in the response

400 Bad Request - The server cannot or will not process the request due to an apparent client error (e.g., malformed

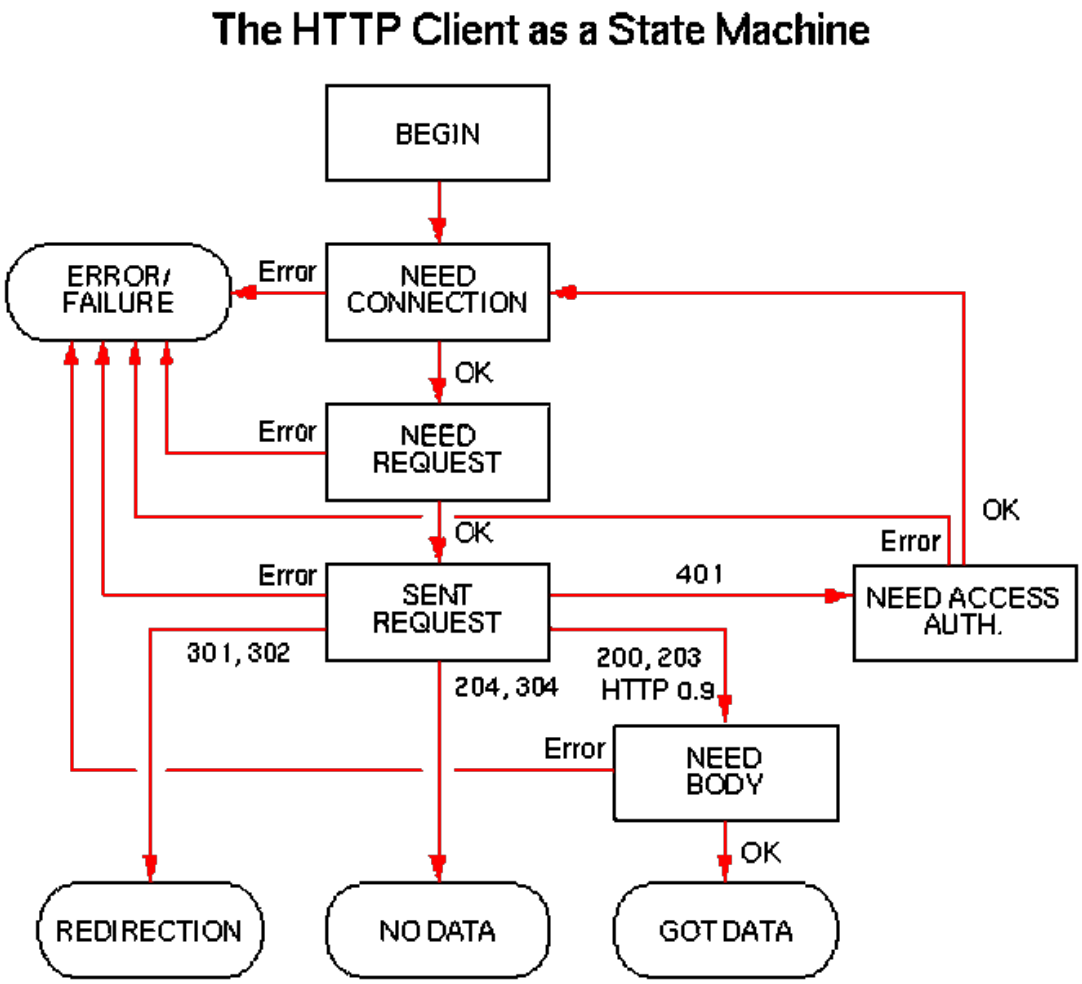
request syntax, size too large, invalid request message framing, or deceptive request routing)

403 Forbidden -The request contained valid data and was understood by the server, but the server is refusing action. This may be due to the user not having the necessary permissions for a resource or needing an account of some sort, or attempting a prohibited action

404 Not Found - The requested resource could not be found but may be available in the future. Subsequent requests by the client are permissible.

500 Internal Server Error - A generic error message, given when an unexpected condition was encountered and no more specific message is suitable

503 Service Unavailable - The server cannot handle the request (because it is overloaded or down for maintenance). Generally, this is a temporary state



A sample conversation between an HTTP client and an HTTP server

Client request

```
GET / HTTP/1.1
Host: www.example.com
```

A client request (consisting in this case of the request line and only one header field) is followed by a blank line, so that the request ends with a double newline, each in the form of a carriage return followed by a line feed. The "Host" field

distinguishes between various DNS names sharing a single IP address,

Server response

```
HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 138
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)
ETag: "3f80f-1b6-3e1cb03b"
Accept-Ranges: bytes
Connection: close

<html>
  <head>
    <title>An Example Page</title>
  </head>
  <body>
    <p>Hello World, this is a very simple HTML document.</
p>
  </body>
</html>
```

The ETag (entity tag) header field is used to determine if a cached version of the requested resource is identical to the current version of the resource on the server.

Content-Type specifies the Internet media type of the data conveyed by the HTTP message

Content-Length indicates its length in bytes.

Accept-Ranges: bytes. This is useful, if the client needs to have only certain portions of a resource sent by the server, which is called byte serving.

Connection: close is sent, it means that the web server will close the TCP connection immediately after the transfer of this response.

Most of the header lines are optional. When Content-Length is missing the length is determined in other ways. Chunked transfer encoding uses a chunk size of 0 to mark the end of the content. Identity encoding without Content-Length reads content until the socket is closed.

A Content-Encoding like gzip can be used to compress the transmitted data.

Web Architecture

can be defined as the conceptual structure of the internet. Types of web architecture include the client-server model and three-tier model.

1. Web Architecture definition
2. Origin of web architecture
3. Types of web architectures

3.1 Client-server model
3.2 Three-tier model
3.3 Service-oriented architectures (SOA)

Web architecture is the conceptual structure of the World Wide Web. The WWW or internet is a constantly changing medium that enables communication between different users and the technical interaction (interoperability) between different systems and subsystems. The basis for this is different components and data formats, which are usually arranged in tiers and build on each other. Overall, they form the infrastructure of the internet, which is made possible by the three core components of data transmission protocols (TCP/IP, HTTP, HTTPS), representation formats (HTML, CSS, XML), and addressing standards (URI, URL).

Origin of web architecture

The world wide web is a concept that was realized in the 1990s so that people and machines could communicate with each other within a certain space. It is used to exchange, distribute, and share information in a network. At that time, the web consisted predominantly of static websites based on HTML , in other words, hypertexts that can be retrieved by a browser. Dynamic websites and distributed web services were added later.

Types of web architectures

The internet is a medium that is constantly changing and expanded by numerous developers, programmers and various consortia such as the W3C (/wiki/W3C). However, the architectures used can be schematically distinguished.

Client-server model

Initially, the web consisted of a two-tiered architecture: clients and servers . Clients and servers shared the tasks and services that the system was supposed to perform. For example, the client may request a service from the server; the server answers the request by providing the service. Retrieving a website using a URL address that directs to a server to load the site in the client's browser is an example of the two-layer model, also known as the client-server model.

The internet protocol family, which now consists of around 500 different network protocols, is usually used as the basis for the WWW, but it usually comprises the TCP/TCP/IP reference model. **Three prerequisites must exist** in the web architecture for the distributed application systems to

communicate with one another:

a) **Representation formats with a fixed standard:**

The most frequently used formats are HTML and CSS ; or XML when machines communicate with one another.

b) **Protocols for data transfer:**

HTTP (Hypertext Transfer Protocol) or HTTPS (Hypertext Transfer Protocol Secure) is used in the web. Other applications, such as mail servers, use SMTP (Simple Mail Transfer Protocol) or POP (Post Office Protocol). Determining the protocols used depends on the application.

c) **The standard for addressing:**

This refers to the URL (Uniform Resource Locator) which is an instance of the more general concept of URI .

Finally, “the web architecture is analogous to the operational structure of application systems for data storage, data transmission, and presentation. When transferred to the web, the web architecture typically consists of database servers that manage the data and resources. They communicate with a client using a transfer protocol that can retrieve the data and view it in a browser. The representation is usually done with HTML and CSS. “

Three-tier model

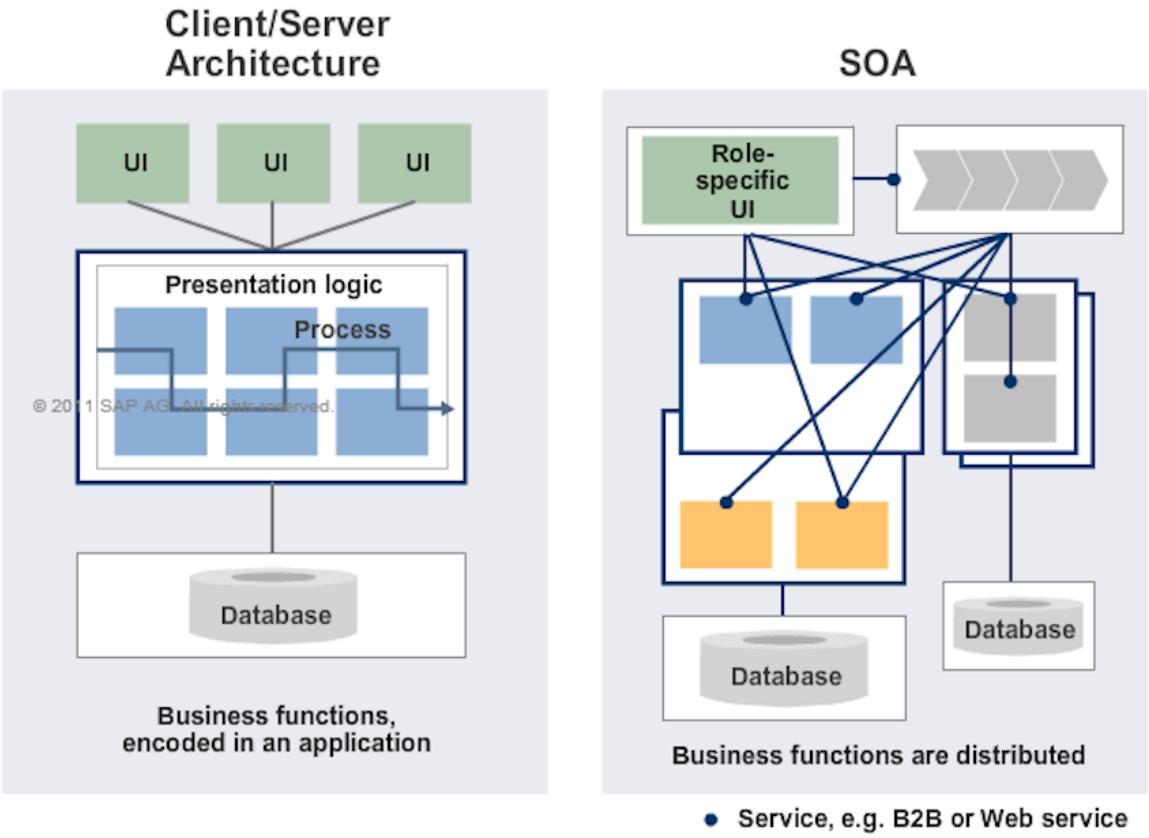
Three-tier models include an application logic between the **client** and the server, which handles the data processing and allows a certain degree of interaction. For example, an **application server** can process data while a **database server** is dedicated solely to data storage. In this way, content can be dynamically loaded and saved. The script language JavaScript is often responsible for the behavior of the client.

There are different programming languages and frameworks to implement three-tier models.

- Hypertext Preprocessor like PHP
- Common Gateway Interface (CGI)
- JavaServer Pages (JSP)
- Active Server Pages (ASP.NET)

- Asynchronous JavaScript and XML (AJAX)

- Microsoft Silverlight
- JavaScript Object Notation (JSON)
- Java applets, JavaScript and VBScript (client-side technologies)



Service-oriented architectures (SOA)

Today the web is used for the networking of globally distributed IT structures. Each IT system can, in turn, consist of subsections whose individual components are linked to one another via a fixed structure or architecture. Think intranet and internal enterprise software. Modern IT and web applications are much more complex than the client-server model.

Distributed web services, which are set up as service-oriented architectures (SOA), offer many functions and modular functional units, which can be supplemented. With SOAs, business processes can be automated by the involved systems

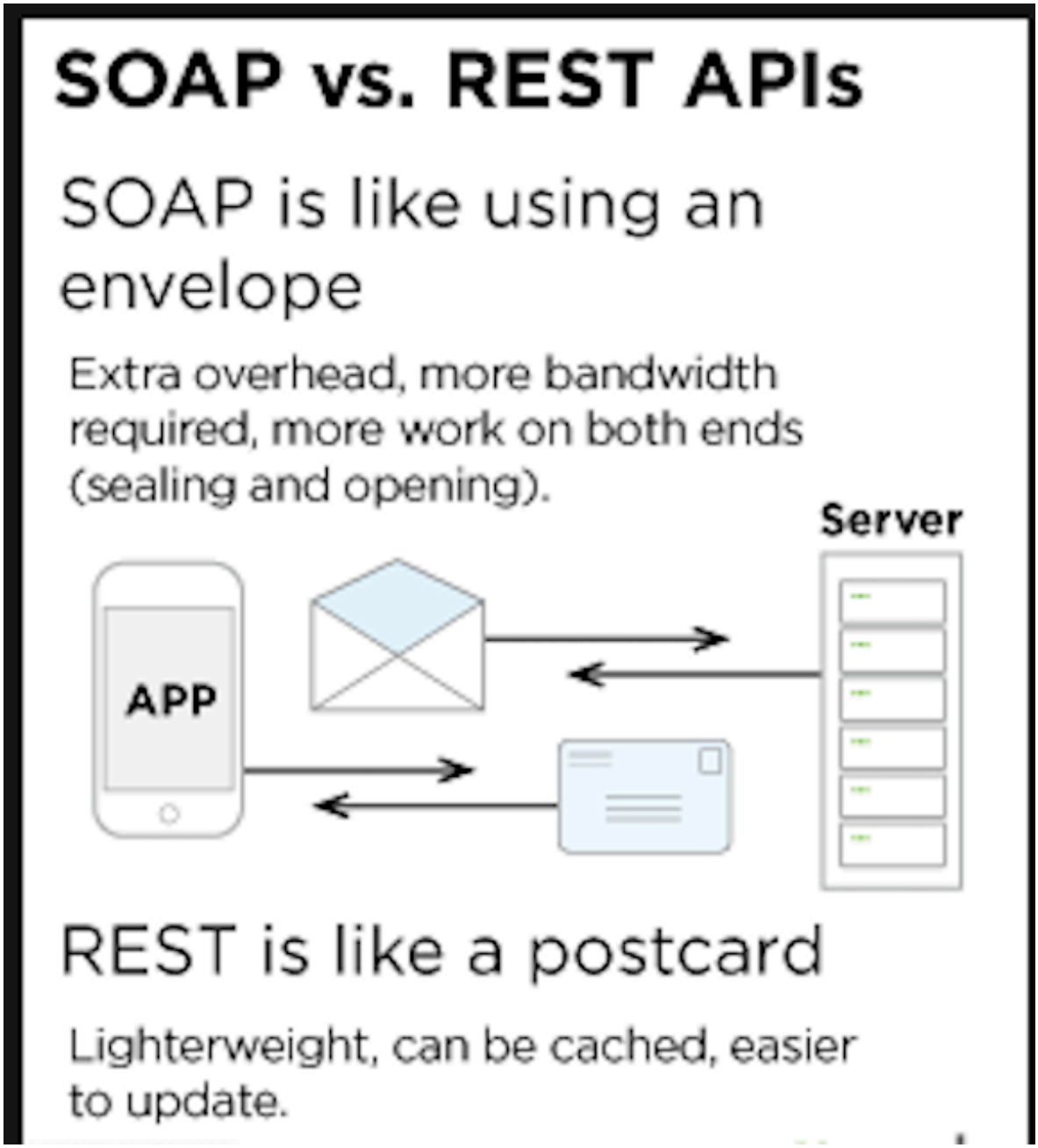
communicating with one another - partly without human intervention - and performing certain tasks. Examples include online banking, e- commerce, e-learning, online marketplaces, and business intelligence applications. These architectures are not only much more complex but can also be modularly extended. They are known as N-tier architectures and have so far been used primarily in the business sector.

There are generally two approaches:

1. Web Services Description Language (WSDL) and Simple Object Access Protocol (SOAP): WSDL is a meta-language for describing network services based on XML, enabling a web service to interpret and execute specific tasks. An interface to a web service can be defined with WSDL. SOAP is also based on XML and allows the control of web services in the form of procedure calls, which are realized with the protocol RPC (remote procedure call). SOAP, WSDL, and XML Schema are often used together.

2. Representational State Transfer (REST): REST is a similar approach used to communicate between machines in distributed systems. It is based on a client-server architecture, but is characterized above all by its uniform interface making REST easy to use with different resources or objects. With the Hypermedia as the Engine of Application State concept, it is also possible to change interfaces during operation, instead of having to redefine them as is the case with WSDL.

#	SOAP	REST
1	A XML-based message protocol	An architectural style protocol
2	Uses WSDL for communication between consumer and provider	Uses XML or JSON to send and receive data
3	Invokes services by calling RPC method	Simply calls services via URL path
4	Does not return human readable result	Result is readable which is just plain XML or JSON
5	Transfer is over HTTP. Also uses other protocols such as SMTP, FTP, etc.	Transfer is over HTTP only
6	JavaScript can call SOAP, but it is difficult to implement	Easy to call from JavaScript
7	Performance is not great compared to REST	Performance is much better compared to SOAP - less CPU intensive, leaner code etc.



The **Internet of Things** or Semantic Web can be considered a current research area in this context. If the Web architecture was represented as an evolutionary timeline, IoT and Semantic Web would be the top of the development. The architectures that are used there are correspondingly complex.