

Big Data Access

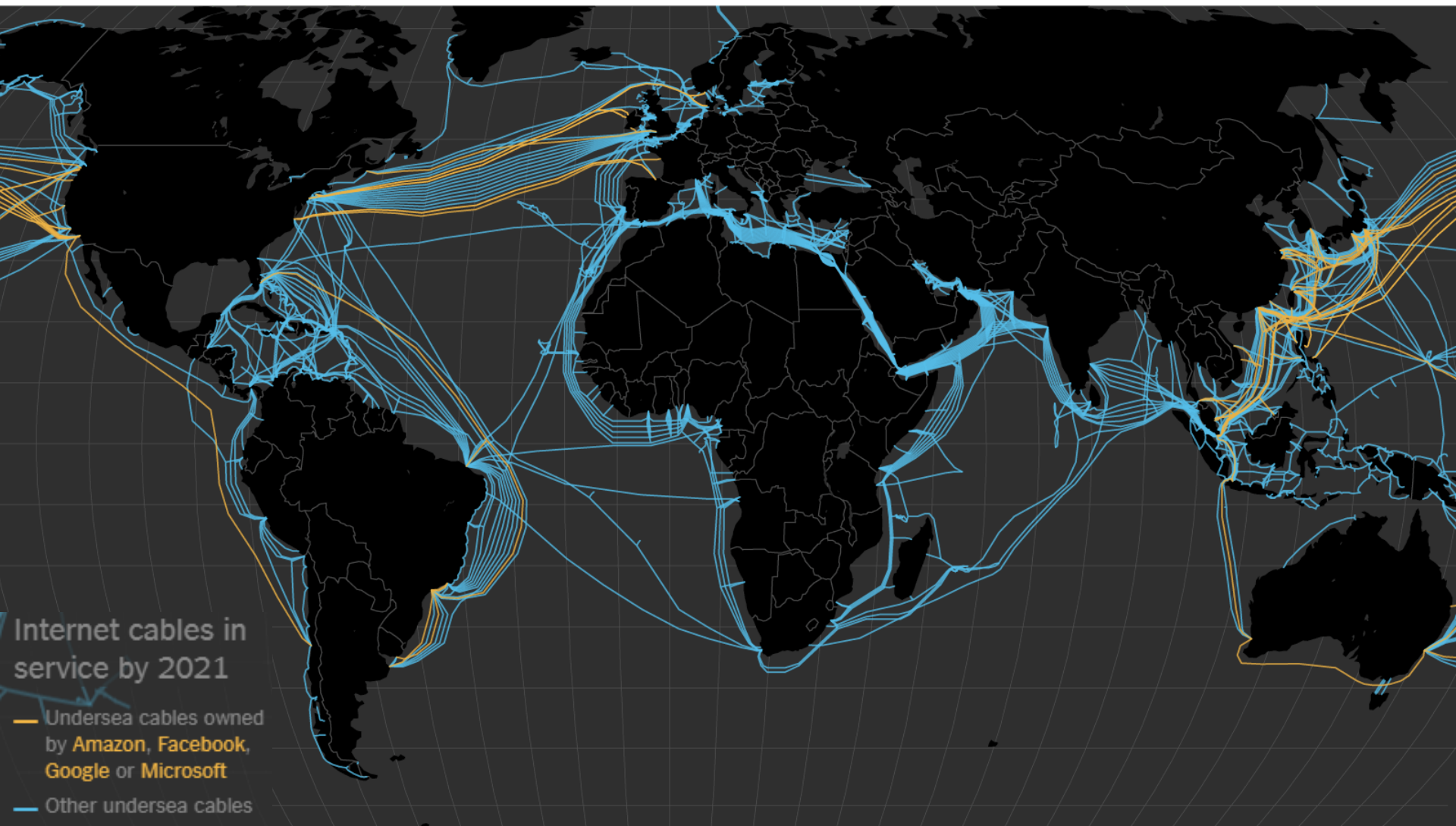
GGE 6505/GGE5405 Introduction to Big Data & Data Science
Winter 2022

An abstract digital cityscape composed of glowing blue cubes and binary code (0s and 1s). The cubes are arranged in a 3D grid, with some cubes having bright blue or red light sources. The background is dark blue, and the overall aesthetic is futuristic and technological.

Outline

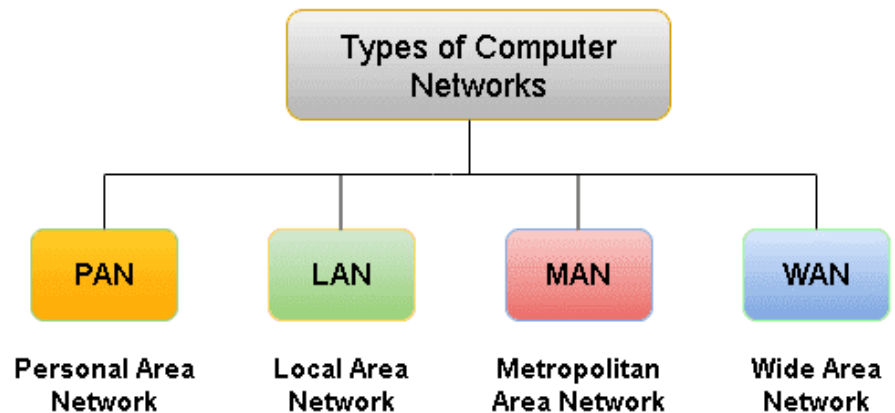
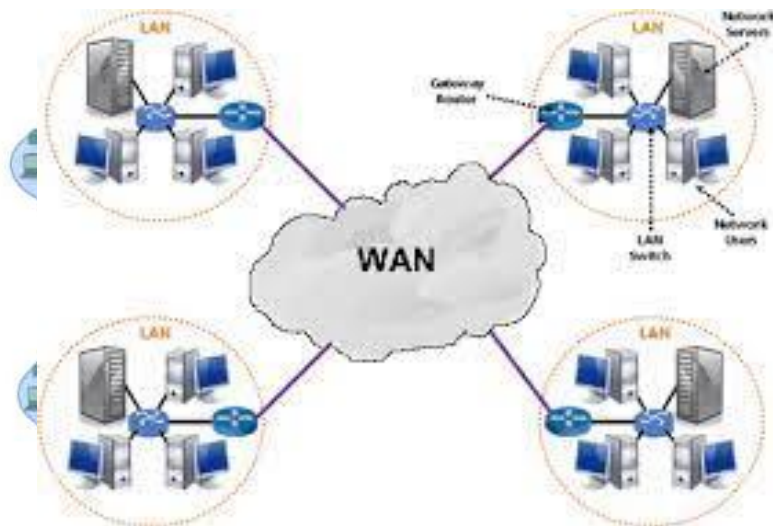
- Network infrastructure
 - Layers
 - Routing
 - Protocols
- API

People think that data is in the cloud, but it's not. It's in the ocean.



Networks

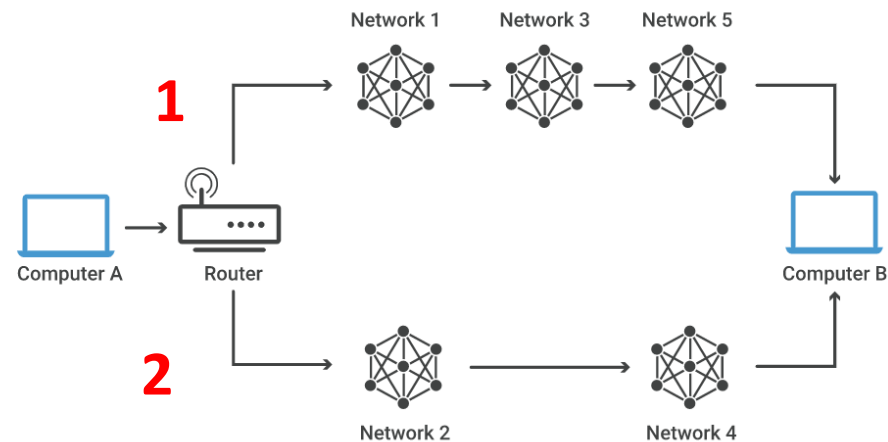
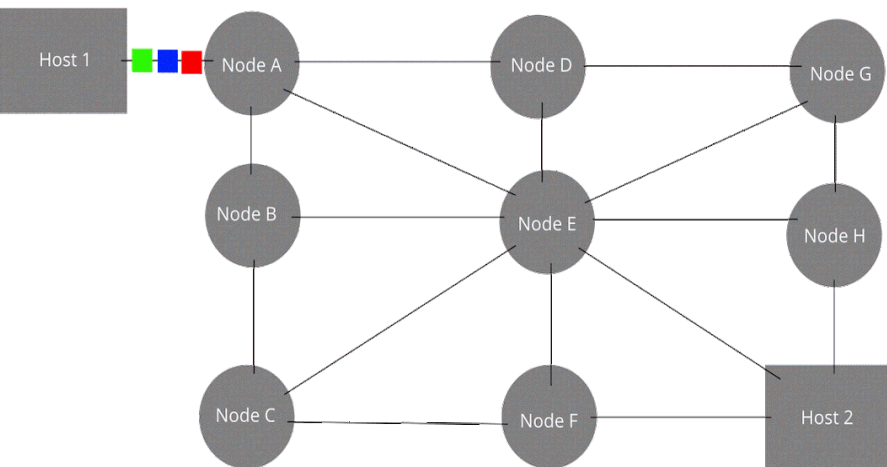
- Networks are the basis for accessing big data.
- Networks are built with a combination of hardware and software to handle distributed resources that are simultaneously working on a single task.
- Networks are defined in terms of purpose and size. From Personal Area Networks to Wide Area Networks



Network Routing

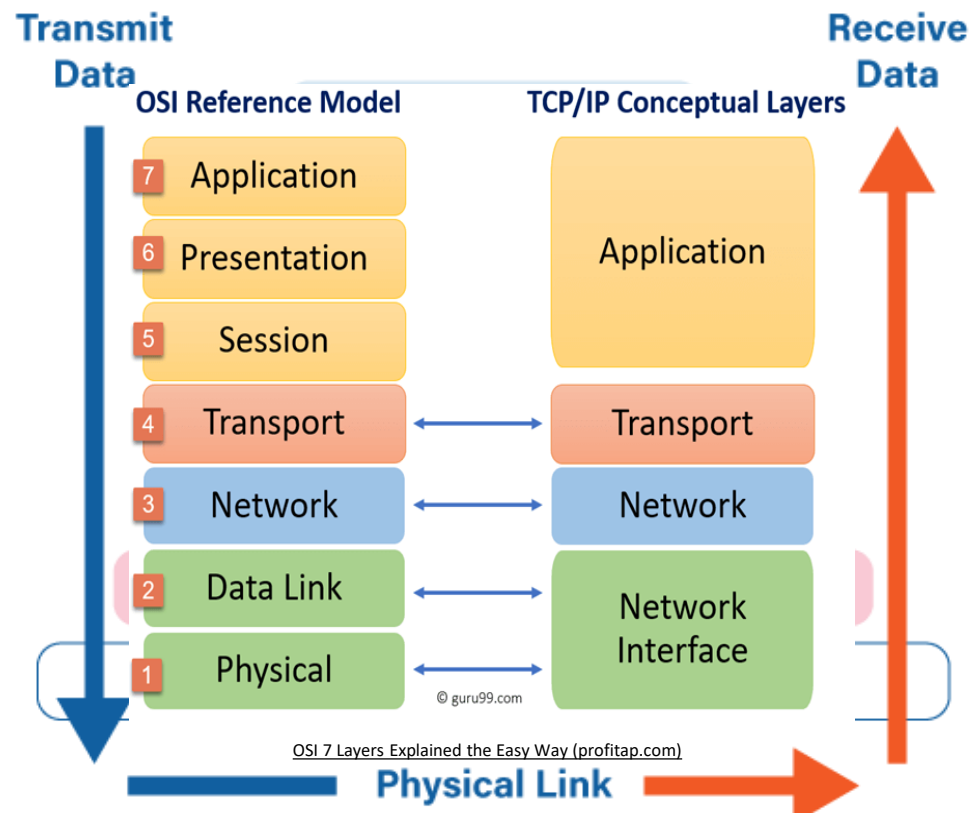
- **Network routing** is the process of selecting a path across one or more networks. The principles of routing can apply to any type of network, from telephone networks to public transportation. In packet-switching networks, such as the Internet, routing selects the paths for Internet Protocol (IP) packets to travel from their origin to their destination.
- **Packet** is a small piece of a larger message. Data sent over computer networks*, like Internet, is divided into packets

The original message is Green, Blue, Red.



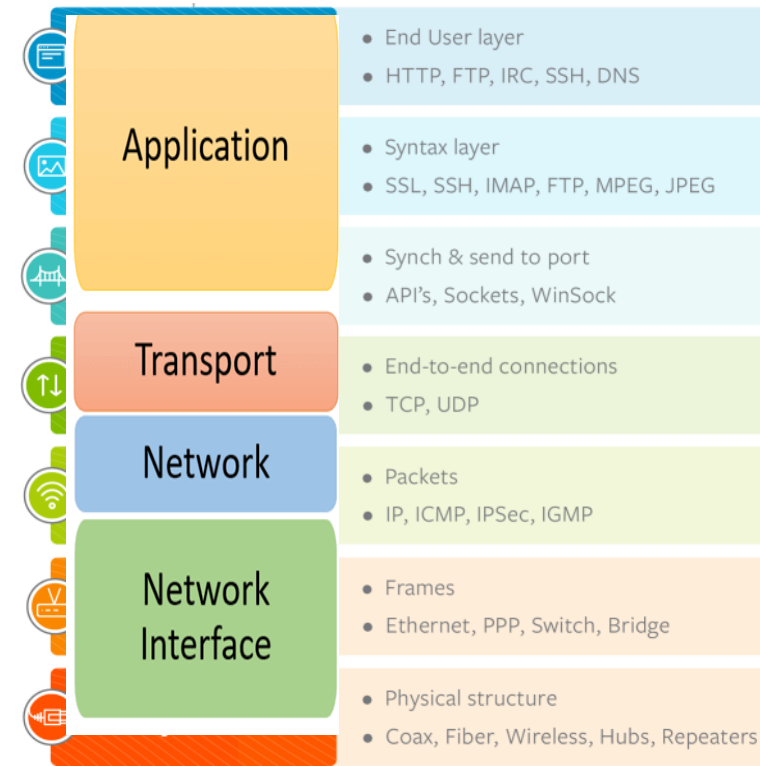
Network Layers

Network-to-network connections are what make the Internet possible. The "network layer" is the part of the Internet communications process where these connections occur, by sending packets of data back and forth between different networks



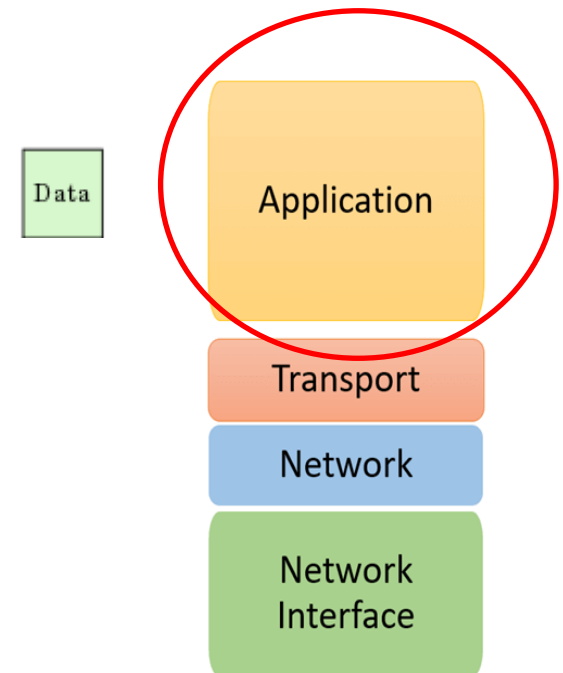
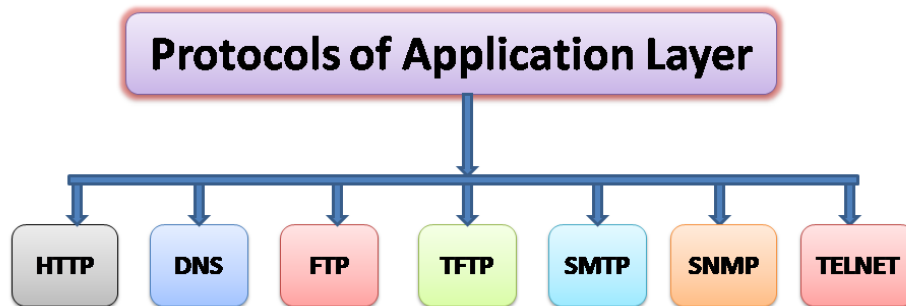
Protocols

- They are formal standards and policies comprised of rules, procedures and formats that define communication between two or more devices over a network.
 - Transmission Control Protocol (TCP)
 - Internet Protocol (IP)
 - User Datagram Protocol (UDP)
 - Simple mail transport Protocol (SMTP)
 - File Transfer Protocol (FTP)
 - Hyper Text Transfer Protocol (HTTP)
 - Hyper Text Transfer Protocol Secure (HTTPS)



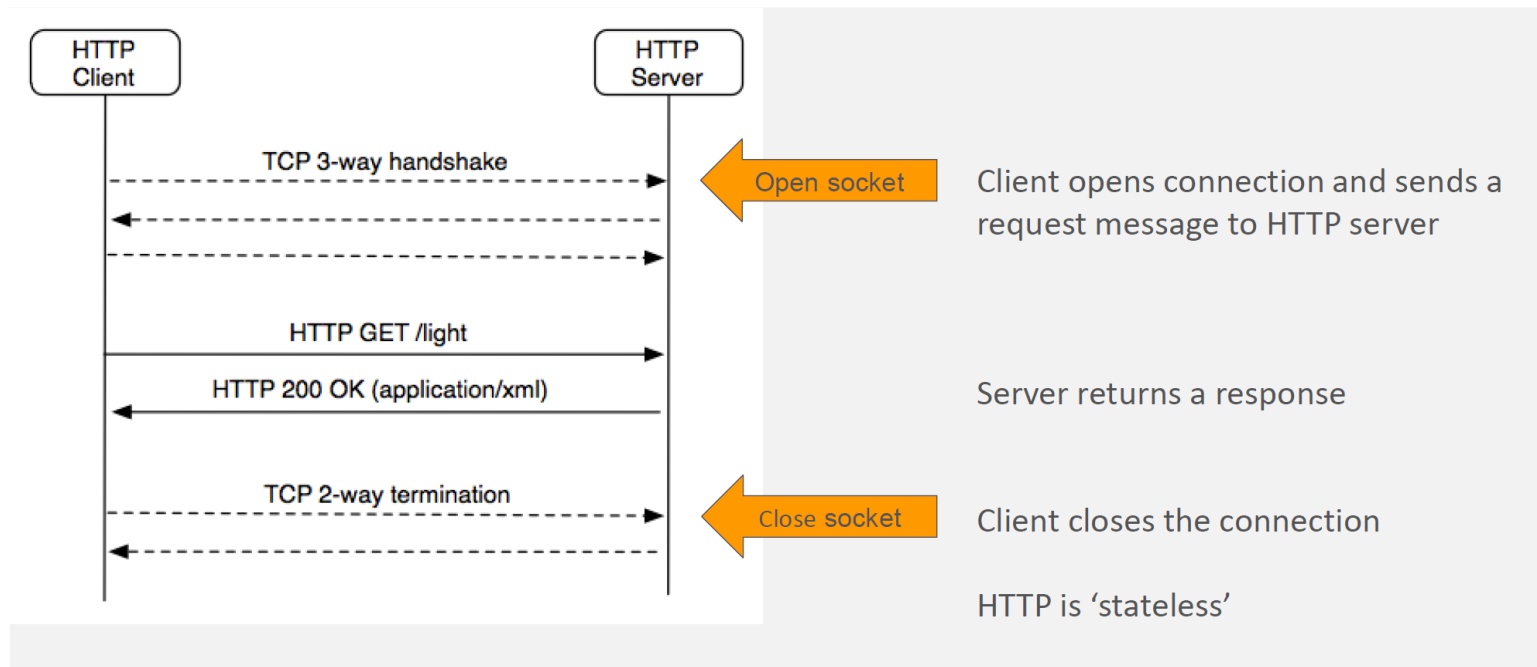
1- Application Layer

- It is responsible for handling high-level protocols, issues of representation.
- This layer allows the user to interact with the application.
- Following are the main protocols used in the application layer:



Request/Response Protocols

- HTTP – Hyper Text Transfer Protocol: it the key protocol to transfer data across the Internet.

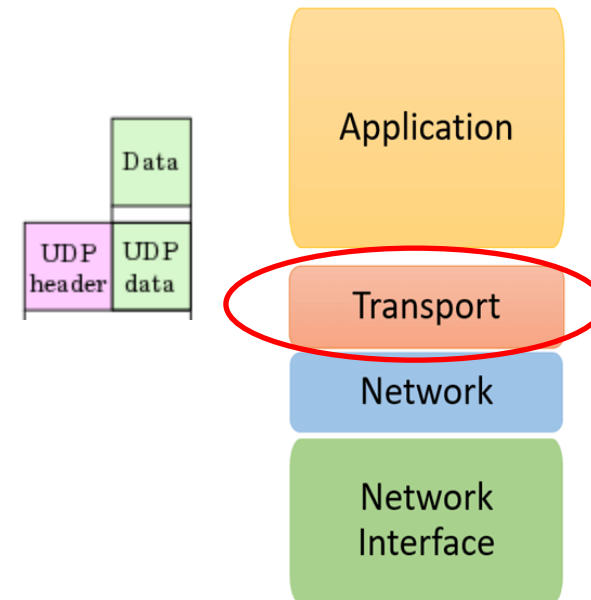
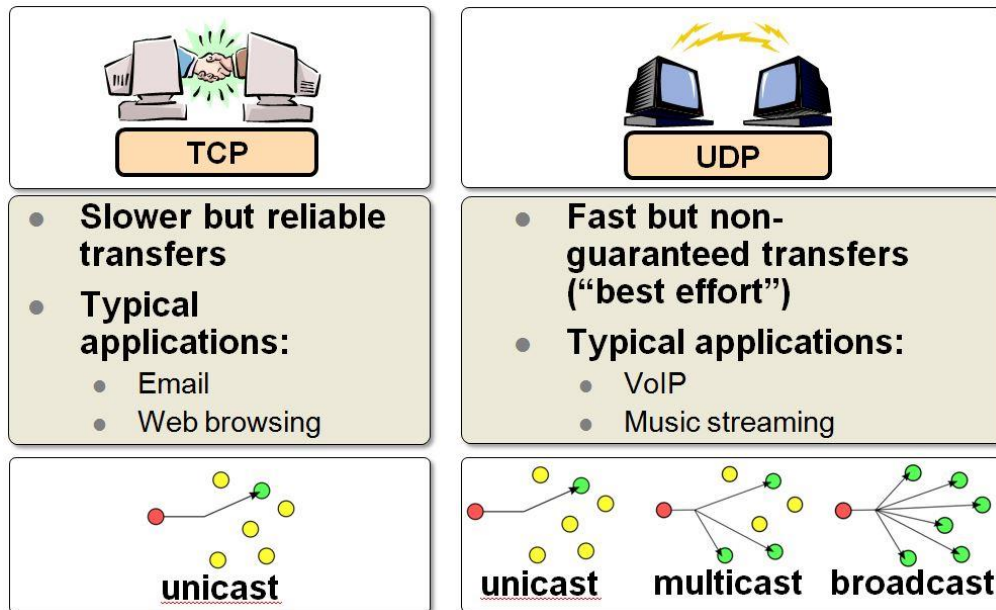


Network Security Protocols

- HTTPS – Hypertext Transfer Protocol Secure: it is a layer on top of HTTP using SSL.
- SSL – Secure Sockets Layer: it allows security by allowing applications to encrypt data than go from a client to a matching server (for example).

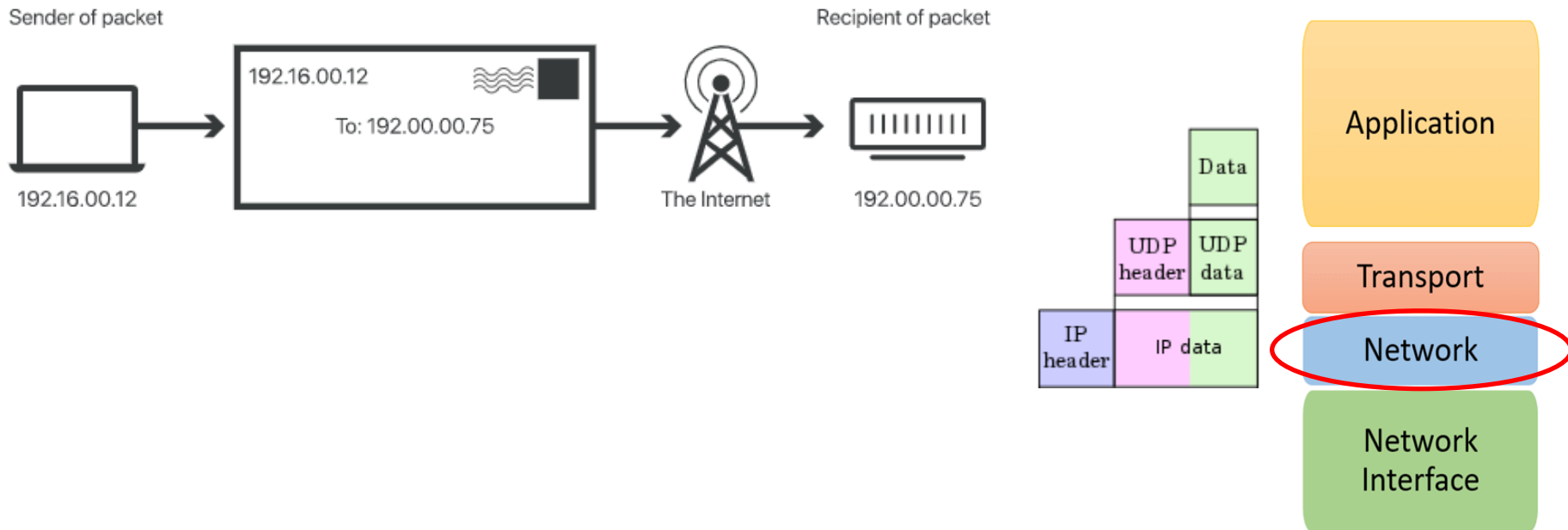
2- Transport Layer

- The transport layer is responsible for the reliability, flow control, and correction of data which is being sent over the network.
- The two protocols used in the transport layer are:
 - User Datagram Protocol (UDP)
 - Transmission Control Protocol (TCP)



3- Network Layer

- The network layer is a portion of online communications that allows for the connection and transfer of data packets between different devices or networks

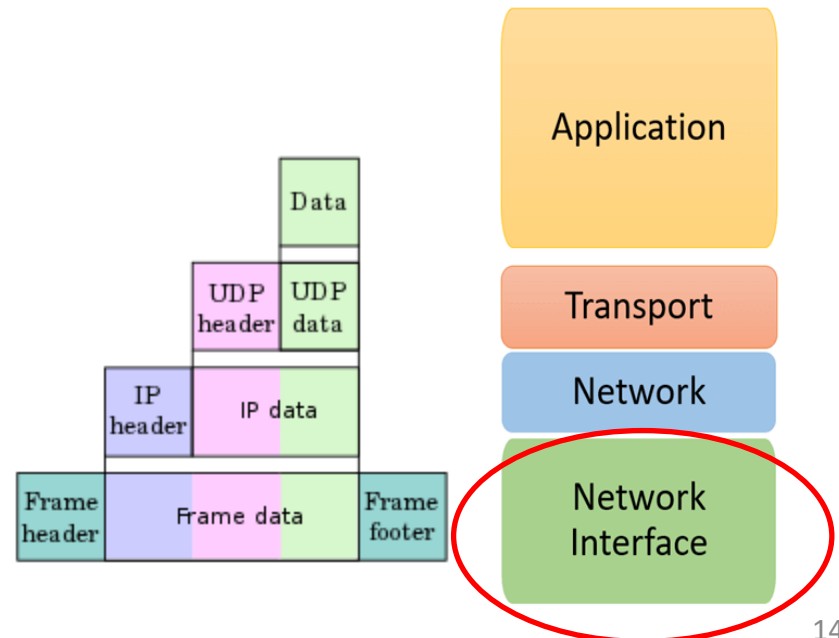


IP ADDRESS



4- Network Interface Layer

- The Network Access Layer is **the lowest layer of the TCP/IP protocol hierarchy**. The protocols in this layer provide the means for the system to deliver data to the other devices on a directly attached network
 - LAN Technologies
 - WAN Technologies



Ways to Access Data

1. Direct download / import of data:

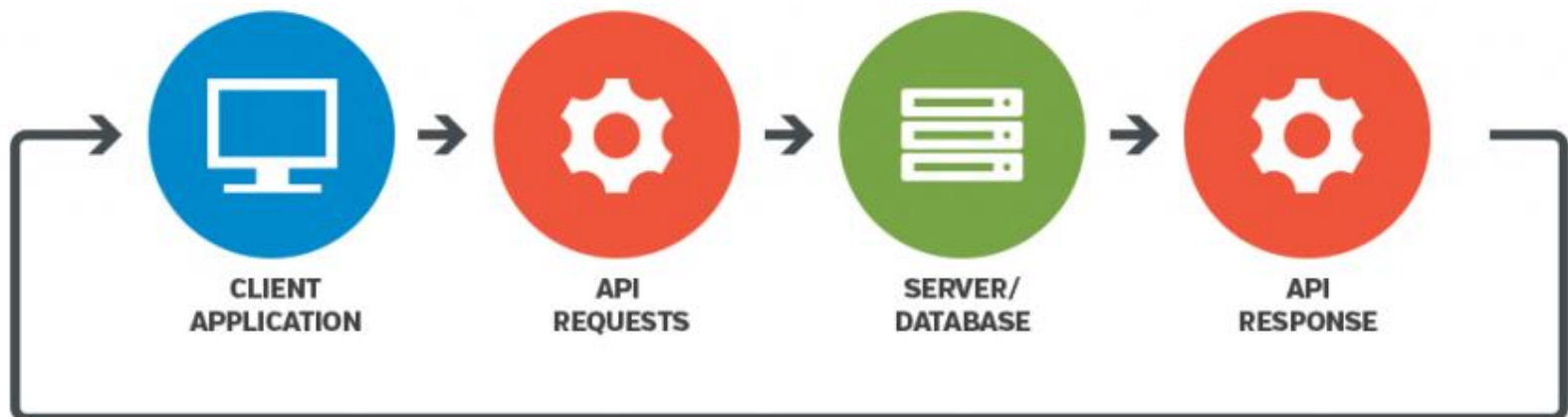
- Data that you directly import into Python using the Pandas function `read_csv`.
- Data that you download by calling a specific URL and using the Pandas function `read_table`, which takes in a url

2. Programming Interfaces (APIs):

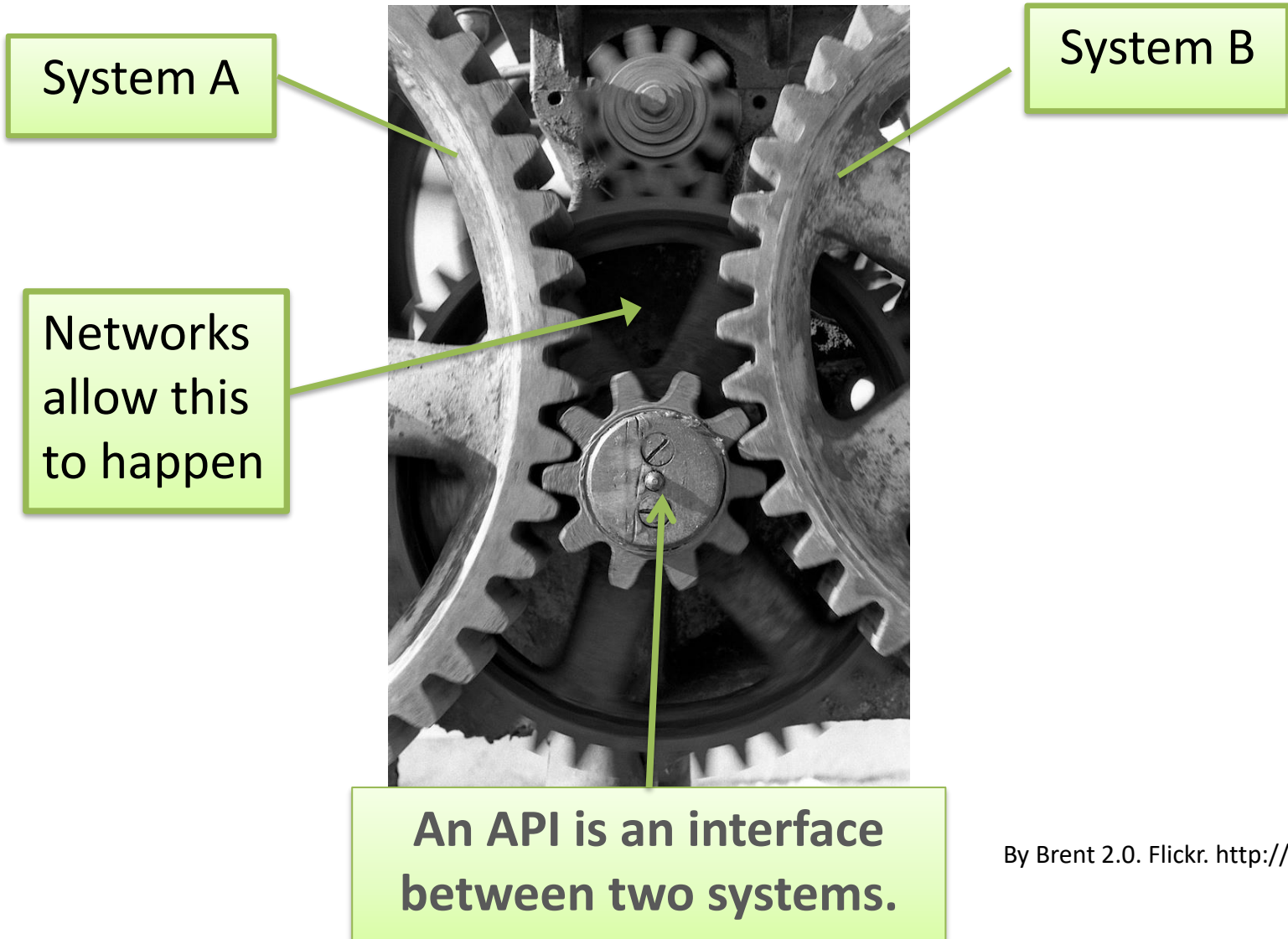
Data that you download using an API, which makes a request to a data repository and returns requested data

Introduction to API

- It stands for **Application Programming Interface**.
- APIs make it easy to share data, computing devices, software between two systems (websites, desktops, smartphones) over a network.
- It uses a set of rules describing how one system can interact with another, and the mechanisms that allow such an interaction to happen.



The API landscape



By Brent 2.0. Flickr. <http://bit.ly/1DexWM0>



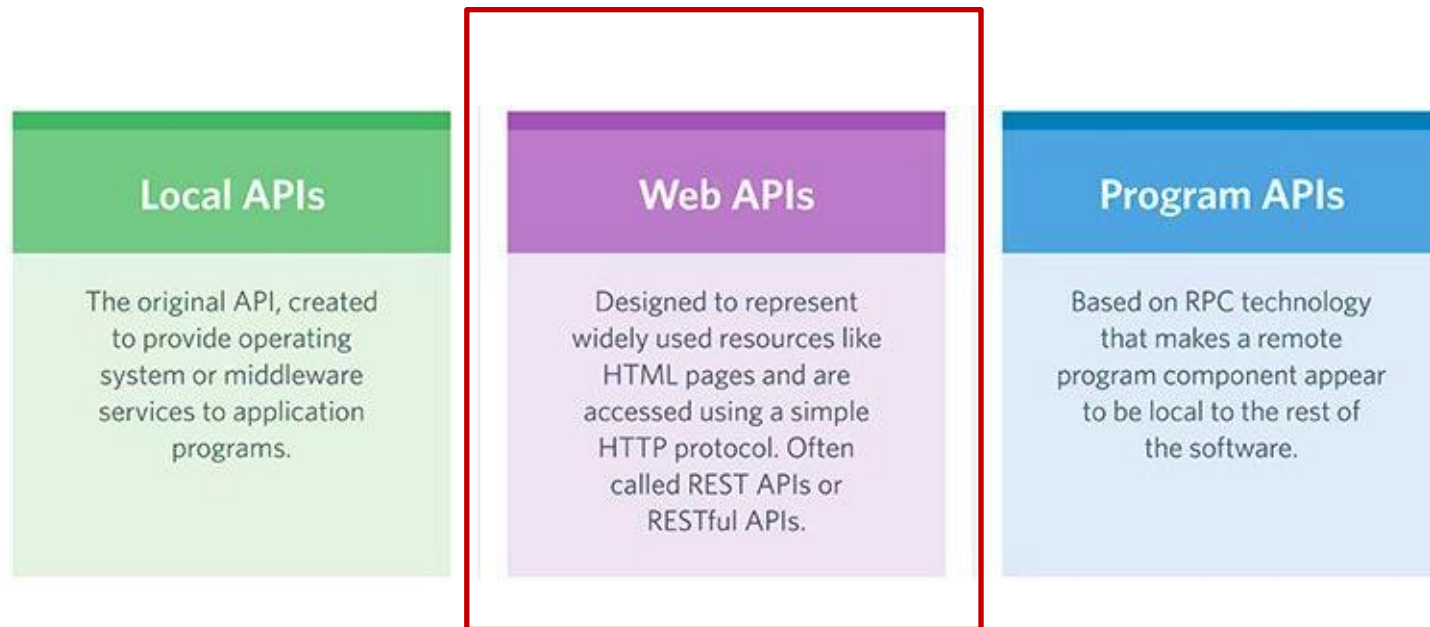
Popular API Examples

1. **Google Maps API's:** Google Maps APIs allows developers to use Google Maps on Webpages using a JavaScript or Flash interface.
2. **YouTube API's:** Google's API lets developers integrate YouTube and functionality into websites or applications. YouTube APIs include the YouTube analytics API, YouTube Data API, YouTube live streaming API, YouTube Player APIs and others.
3. **Planet APIs:** It is used by developers allows you to search Planet's complete catalog of data. Other APIs are analytics, base maps, tile services, etc.
4. **Twitter APIs:** Twitter offers two APIs, the REST API allows developers to access core Twitter data and the search API provides methods for developers to interact with twitter search and trends data.
5. **Kaggle APIs:** Create Datasets, Notebooks, and connect with Kaggle. The Kaggle API and CLI tool provide easy ways to interact with Notebooks on Kaggle.

Types of API

Public API: is publicly available for all developers to access

Private API: is only exposed to internal developers therefore the API publishers have total control over what and how applications are developed.



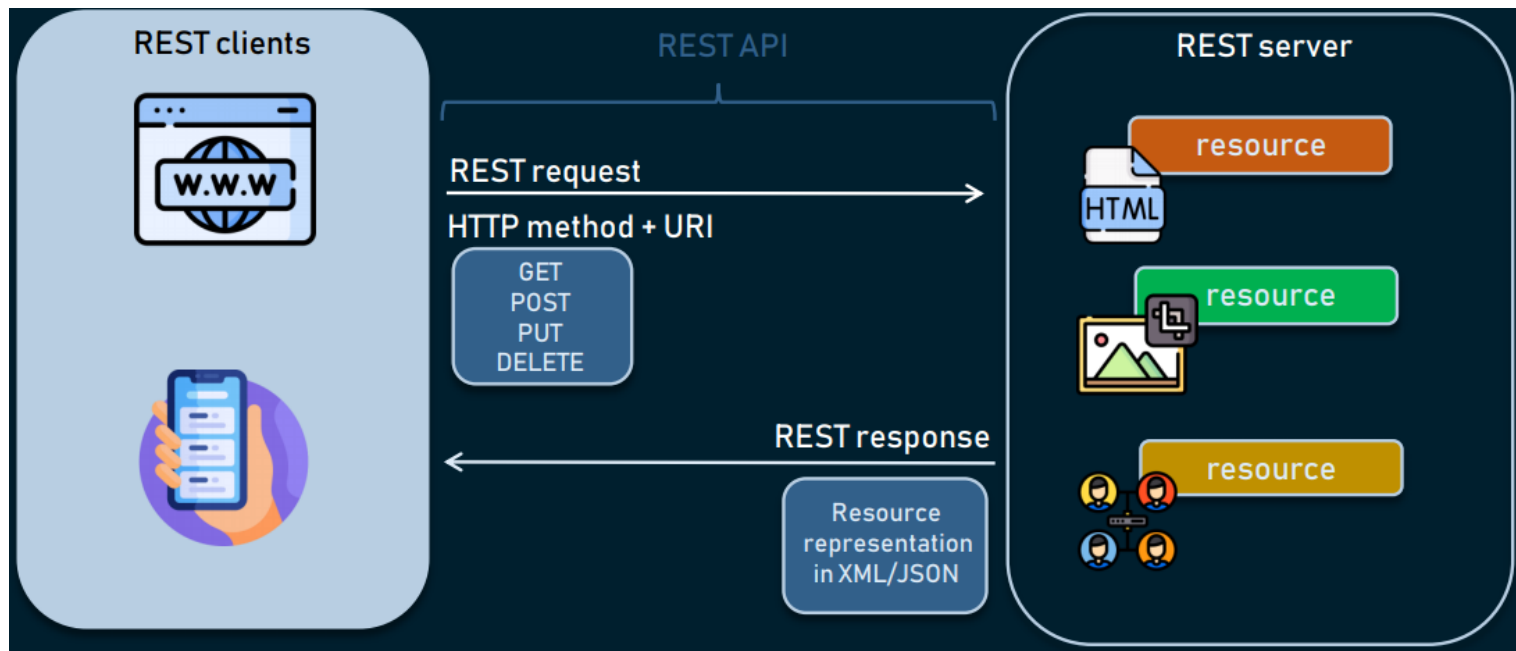
Types of API

SOAP (Simple Object Access Protocol)	REST (REpresentational State Transfer)
Protocol	Architecture
Function driven	Data driven
Requires advanced security, but can define it, too	Relies on underlying network which can be less secure
Needs more bandwidth	Only needs minimum bandwidth
Stricter rules to follow	Easier for developers to suggest recommendations
Cannot use REST	Can use SOAP
Only works in XML	Works in different data formats such as HTML, JSON, XML, and plain text
Supports HTTP and SMTP protocols	Only requires HTTP

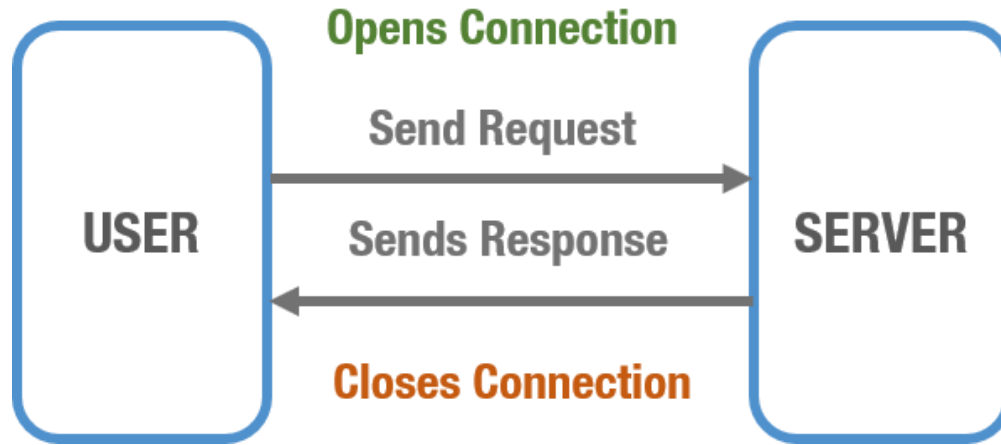
- Rest API
- Streaming API

REST API

- **REST API** (Representational State Transfer) is designed to take advantage of existing protocols.
- REST APIs are used by such sites as Amazon, Google, LinkedIn and Twitter.
- Uses existing features of the HTTP protocols to perform caching and security enforcement.



REST API



- After the response is sent back to the user, the connection closes only to be re-opened when the user makes another request to the API.
- REST APIs are perfect for users wanting a “snapshot” of data where the information does not change very frequently.
- In Twitter, a user is allowed 350 requests per hour, up to 1500 tweets per request.

HTTP Requests

To make a valid request, the client needs to include four things:

1. URL

2. Method

- **GET** - Asks the server to retrieve a resource
- **POST** - Asks the server to create a new resource
- **PUT** - Asks the server to edit/update an existing resource
- **DELETE** - Asks the server to delete a resource

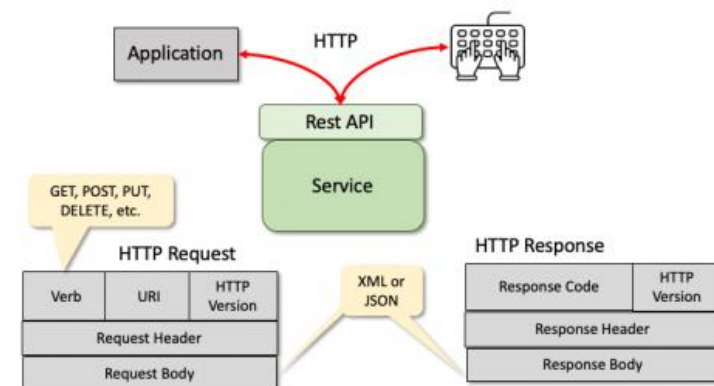
3. Headers

4. Body

Representing Data

The data that you access programmatically may be returned in one of two main formats:

- 1. Tabular Human-readable file:** Files that are tabular, including CSV files (Comma Separated Values) and even spreadsheets (Microsoft Excel, etc.). These files are organized into columns and rows and are “flat” in structure rather than hierarchical.
- 2. Structured Machine-readable files:** Files that can be stored in a text format but are hierarchical and structured in some way that optimizes machine readability.
 - JSON (JavaScript Object Notation)
 - XML (Extensible Markup Language).



JSON (JavaScript Object Notation)

- Many new APIs have adopted JSON as a format because it's built on the popular JavaScript programming language.
- JSON is a very simple format (Associative Array) that has two pieces: **keys** and **values**.
 - **Keys** represent an attribute about the object being described. A pizza order can be an object. It has attributes (keys), such as crust type, toppings, and order status.
 - These attributes have corresponding **values** (thick crust, pepperoni, and out-for-delivery).

Pizza Order in JSON

- Let's see how this pizza order could look in JSON:

key *value*

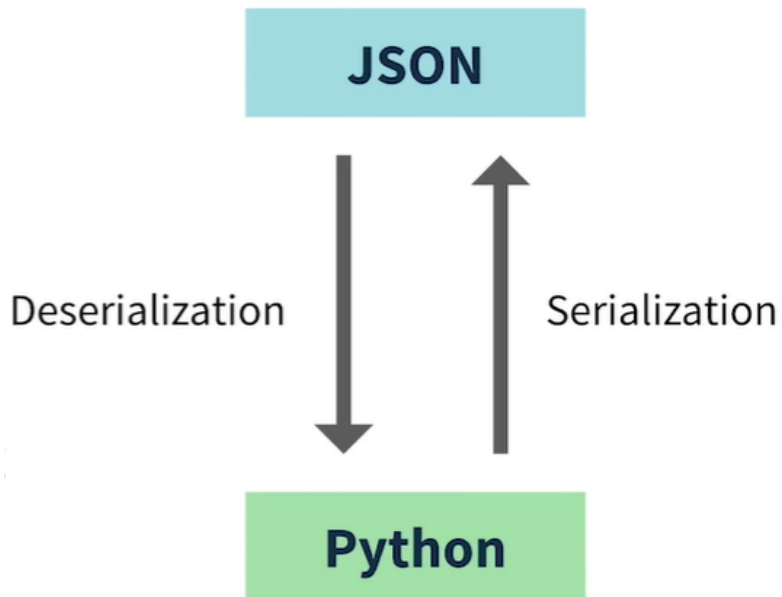
 ↘ ↙

```
{ "crust": "original" }
```

```
{ "crust": "original",  
  "toppings": ["cheese", "pepperoni",  
               "garlic"],  
  "status": "cooking",  
  "customer": { "name": "Brian",  
                 "phone": 573111111 } }
```

JSON in Python

```
import json
```



`dump()`

Write data to
a file-like
object in
JSON format

`dumps()`

Write data to
a string in
JSON format

JSON Types in Python

Python object



Python	JSON
dict	object
list, tuple	array
str	string
int, long, float	number
True	true
False	false
None	null

JSON



Working With JSON Data in Python – Real Python



```
import json
```

```
[7] pizza = {}  
    pizza['Brian']= {  
  
        'crust': 'original',  
        'toppings' : ['cheese', 'pepperoni', 'garlic'],  
        'status' : 'cooking'  
    }  
  
    pizza['Bob']={  
  
        'crust': 'thin',  
        'toppings' : ['cheese', 'olive', 'mashroom'],  
        'status' : 'cooking'  
    }
```

```
[8] jn = json.dumps(pizza)
```

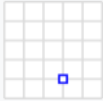
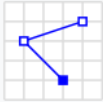
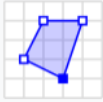
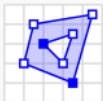
```
print(jn)
```

```
{"Brian": {"crust": "original", "toppings": ["cheese", "pepperoni", "garlic"], "status": "cooking"}, "Bob": {"crust": "thin", "toppings": ["cheese", "olive", "mashroom"], "status": "cooking"}}
```

GeoJSON

- GeoJSON is an open standard format designed for representing simple geographical features, along with their non-spatial attributes. It is based on the JSON format.
- Check : geojson.io

```
{ "type": "Feature",  
  "geometry": {  
    "type": "Point",  
    "coordinates": [125.6, 10.1] },  
  "properties": {  
    "name": "Dinagat Islands"  
  }  
}
```


Type	Examples	
Point		<pre>{ "type": "Point", "coordinates": [30.0, 10.0] }</pre>
LineString		<pre>{ "type": "LineString", "coordinates": [[30.0, 10.0], [10.0, 30.0], [40.0, 40.0]] }</pre>
Polygon		<pre>{ "type": "Polygon", "coordinates": [[[30.0, 10.0], [40.0, 40.0], [20.0, 40.0], [10.0, 20.0], [30.0, 10.0]]] }</pre>
		<pre>{ "type": "Polygon", "coordinates": [[[35.0, 10.0], [45.0, 45.0], [15.0, 40.0], [10.0, 20.0], [35.0, 10.0]], [[20.0, 30.0], [35.0, 35.0], [30.0, 20.0], [20.0, 30.0]]] }</pre>

XML (Extensible Markup Language)

- XML has been around since 1996.
- With age, it has become a very mature and powerful data format.
- Like JSON, XML provides a few simple building blocks that API makers use to structure their data.
- The main block is called a root **node** which contains child **nodes**.

node opening tag value node closing tag

 \ ↓ /

 <crust>original</crust>

Pizza Order in XML

- Let's see what our pizza order might look like in XML:

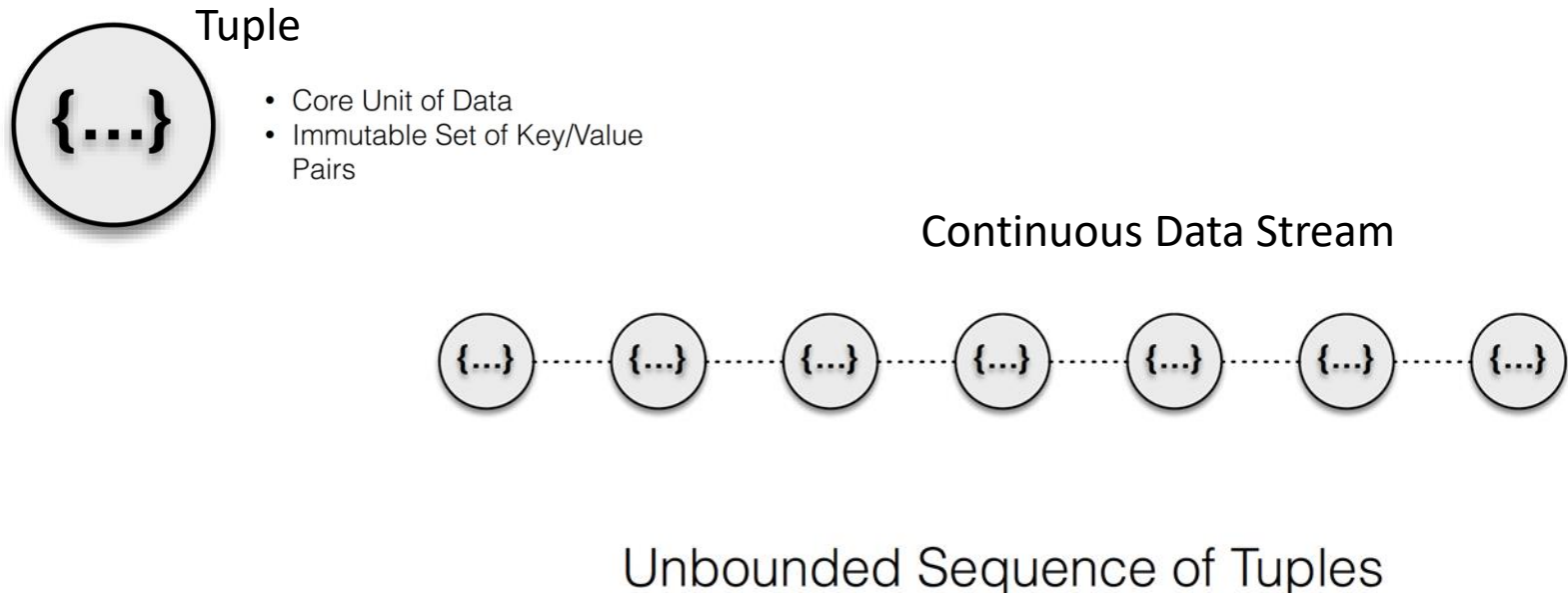
```
<order> <crust>original</crust> <toppings>  
<topping>cheese</topping> <topping>pepperoni</topping>  
<topping>garlic</topping> </toppings>  
<status>cooking</status> </order>
```

- Notice how in XML, every item in the list of toppings is wrapped by a node. You can see how the XML format requires a lot more text to communicate than JSON does.
- JSON is faster than XML because of less verbose.

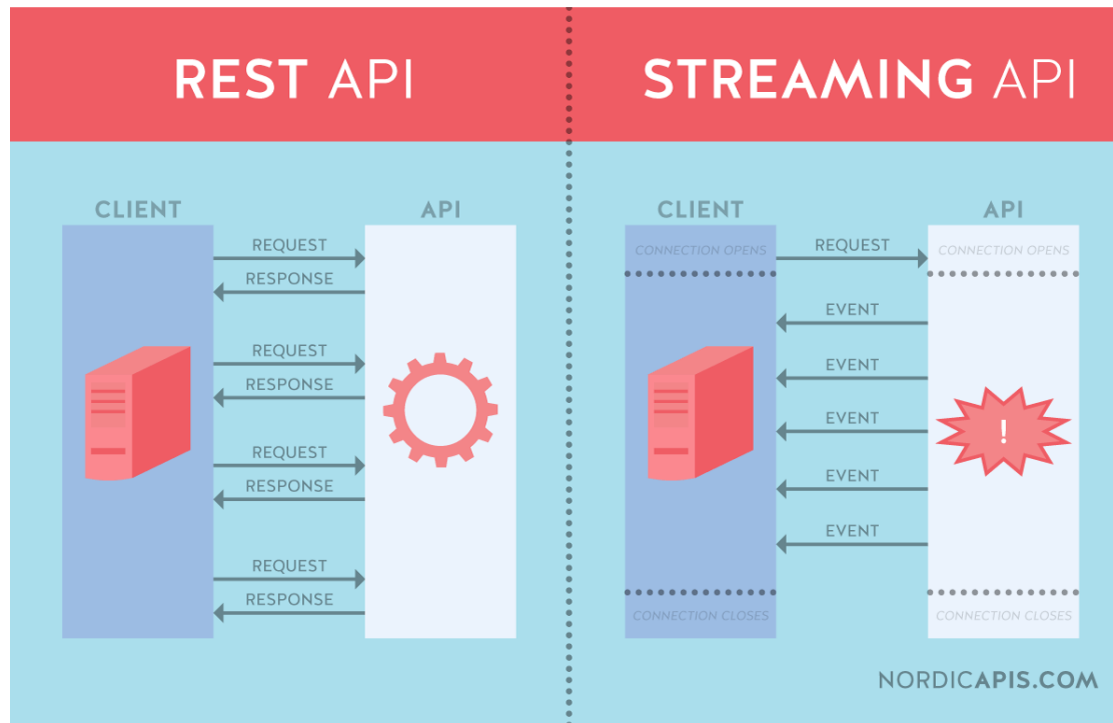
- Rest API
- Streaming API

Streaming API

- It is usually used for reading data streams.
- Uses network protocols such as Webhooks, WebSockets, XMPP, SMTP, gRPC and MQTT.



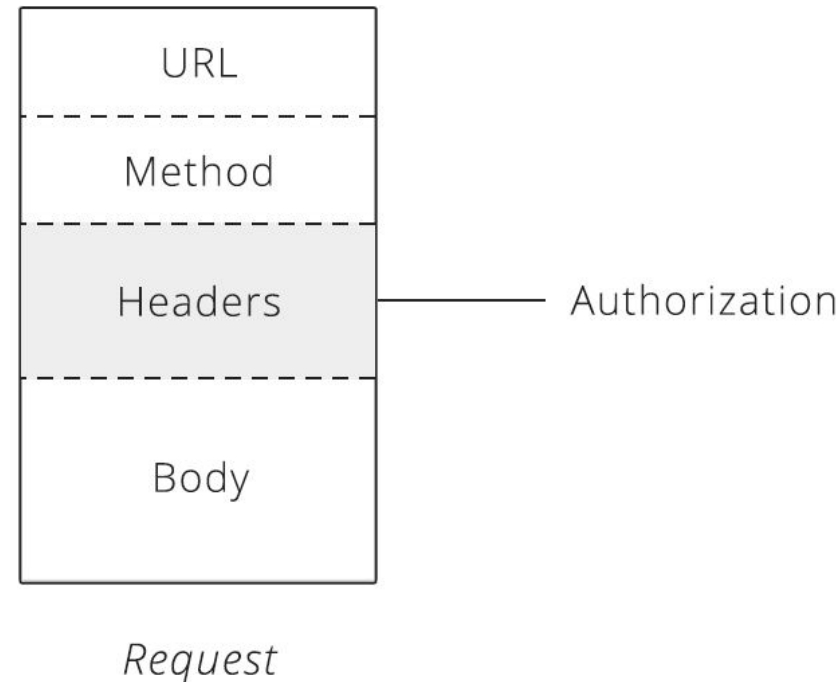
Streaming API



- It maintains a persistent connection that continuously sends updated data to the user until the connection is terminated.
- Streaming APIs are perfect when a user needs to consume a constant flow of rapidly updating live data.
- The server repeatedly sends responses back with updated information until the connection is eventually closed by the user.

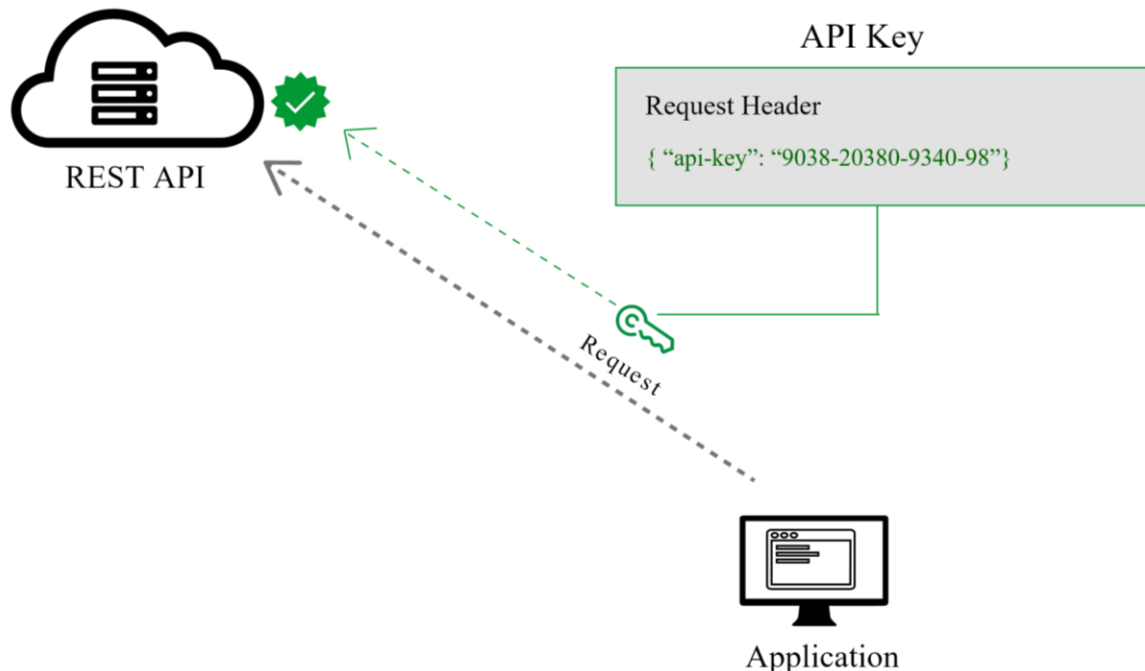
Basic Authentication

- Basic Auth only requires a username and password. The client takes these two credentials, smooshes them together to form a single value, and passes that along in the request in an HTTP header called Authorization.



API Key Authentication

- **Authentication:** process of the client proving its identity to the server
- **Credentials:** secret pieces of info used to prove the client's identity (username, password...)
- **Basic Auth:** scheme that uses an encoded username and password for credentials
- **API Key Auth:** scheme that uses a unique key for credentials
- **Authorization Header:** the HTTP header used to hold credentials



API and Python

Python Libraries for API

- Requests.
- Faster Than Requests.
- PycURL.
- Flask.
- Tornado.
- FastAPI.
- ...

Requests Library in Python



- The library is used for making API requests.
- Requests support all types of HTTP methods as well as advanced HTTP features like authentication, SSL handling, session cookies, and more.



#GitHub Data

```
import requests
```

```
res = requests.get('https://api.github.com/users/nasrineshraghi')  
data = res.json()  
print(data)
```

```
{'login': 'nasrineshraghi', 'id': 38828636, 'node_id': 'MDQ6VXNlc
```