Notes

What is a Server?

- If we just look at the word itself, `server` is the one that serves something.
- The system, that is making the request for something, is called the `client`
- `Server` accepts the request from the `client` and sends the response based on the request.

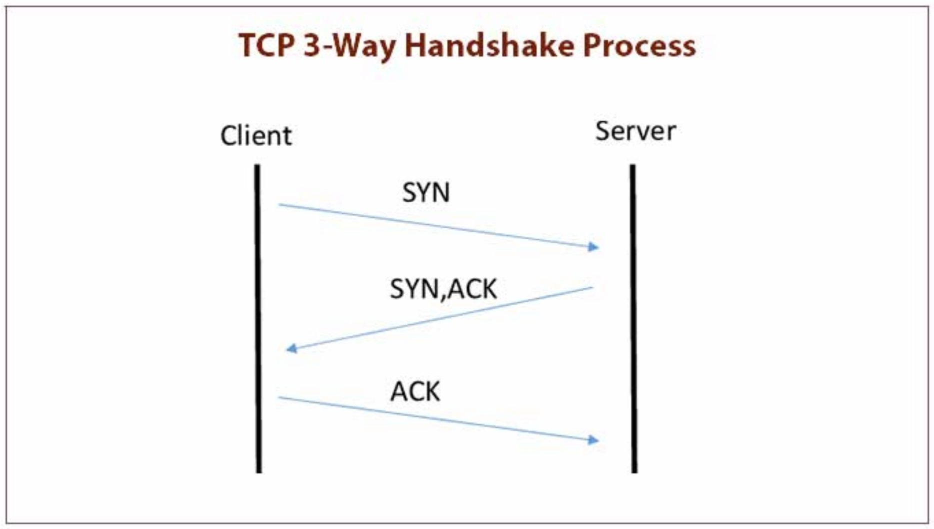
HTTP

- It stands for `Hyper Text Transfer Protocol` .
- Set of Protocol or rules that are required to communicate between client and server.
- **HTTPS:-** It is exactly the same with an added layer of security, it is achieved by SSL(Secure Socket Layer).

3 Way Handshake

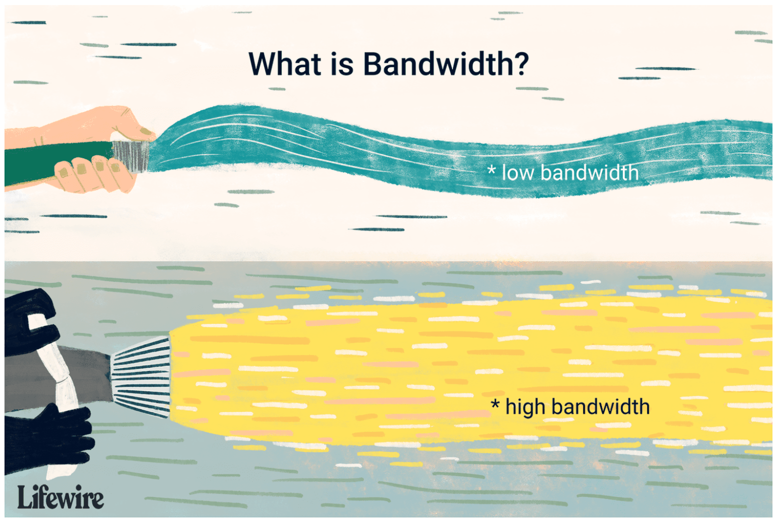
A 3 Way handshake takes place between the client and server, before the actual cycle starts

- First the client expresses the intent to the server.
- Server Acknowledges the client's intent.
- Client will tell what it actually needs.



Bandwidth

- Bandwidth is **the data transfer capacity of a computer network.**



HTTP Verbs/Methods

- **GET:** To read something from the server.
- **PUT⇒Modify:** Will replace the whole thing.
- **PATCH⇒Modify:** Will modify one specific thing in the whole thing.

- **Delete:** To delete something on the server.
- **POST:** To post/add/sent something to the server.
- We can achieve CRUD through this.

Creating First Server

- Create a node project by `npm init -y` .
- Create a file named `index.js` .
- Now for creating the server we can use the inbuilt `http` module of node.

```
const http = require("http")

const server = http.createServer((request,response) => {
    if(request.url === "/"){
        response.end("Hello")
    } else if(request.url === "/reports"){
        response.end("Here are the reports")
    } else if(request.url === "/data"){
        response.end("Data....")
    }
})

server.listen(4500,() => {
    console.log("Listening on the port 4500")
})
```

- We have programmed our server to give the response as per the request made.
- Whenever we make any changes in server we have to re run the server.

`.end` vs `.write`

💡 **Here we are using `.write`, Now the client will not know that the response has been ended and it will keep on loading the page, that is why we have to use `.end`, so that the client knows that response has been ended.**

Invalid End Point

- What if the user is making a request to an invalid end point, then we have to take care of that as well while programming our server.

```
const http = require("http")

const server = http.createServer((req,res) => {
    if(req.url === "/"){
        res.end("Hello")
    } else if(req.url === "/reports"){
        res.end("Here are the reports")
    } else if(req.url === "/data"){
        res.end("Data....")
    } else{
        res.end("Invalid End Point")
    }
})

server.listen(4500, () => {
    console.log("Listening on the port 4500")
})
```

Send Data from a file

- We can also send other things as a response as well.
- Let us try sending data which is inside a file as a response.
- Create a `text.txt` with some dummy data inside it.

```
const http = require("http")
const fs = require("fs")

const server = http.createServer((req,res) => {
    if(req.url === "/data"){
        fs.readFile("./text.txt", {encoding:"utf-8"}, (err,data
            if (err) {
                res.write("No data\n")
                res.end(err)
            } else {
                res.end(data)
            }
        })
    }
})

server.listen(4500, () => {
    console.log("Listening on the port 4500")
})
```

💡 **The above code will result in getting data as the response of the request is made at `/data` endpoint.**

Headers

- **What are headers?** ⇒ It just gives more information about the request or response.
- We also want to send a header as a response.
- This is just to specify what kind of response we are getting.
- Let's pass a header as well.

```
const http = require("http")
const fs = require("fs")

const server = http.createServer((req,res) => {
    if(req.url === "/"){
        res.setHeader("Content-type", "text/html") //Header to s
        res.end("<h1>Hello Guys!!</h1>")
    }
})

server.listen(4500, () => {
    console.log("Listening on the port 4500")
})
```

Research Work:

💡 **Stream:** https://nodejs.org/dist/latest-v18.x/docs/api/stream.html
Also research about what exactly this is.

HTTP: https://nodejs.org/dist/latest-v18.x/docs/api/stream.html