

1. Observe el código de los programas TCP proporcionados en la Guía de Laboratorio y notara dos situaciones: a) el cliente no envía ningún dato al servidor, solo realiza la conexión y recibe un mensaje de respuesta, b) el servidor, siempre retorna el mismo mensaje "Hola Mundo". Ahora deberá modificar ambos programas de manera que las nuevas versiones se ajusten a las siguientes especificaciones:

a) El programa cliente deberá recibir como argumento el mensaje a enviar, por lo que la ejecución de su programa cliente debería ser invocada con: `nodejs programa mensaje`.

b) El programa servidor deberá retornar la cadena recibida en mayúsculas.

=====SERVIDOR=====

En el servidor agregamos ese bloque de código que pone al servidor en espera de algún mensaje por el cliente y posteriormente reenvía este mismo mensaje escrito en mayúscula.

```
var net = require('net');

var server = net.createServer(function(connection) {
  console.log('cliente conectado');
  connection.on('data', function(mensaje){
    connection.write(mensaje.toString().toLocaleUpperCase());
  });
  connection.on('end', function() {
    console.log('cliente desconectado');
  });
});

server.listen(8080, function() {
  console.log('servidor esta escuchando');
});
```

=====CLIENTE=====

Para el lado del cliente inicialmente para poder recibir el mensaje como argumento podemos usar un bucle for para obtener todas las palabras que se nos envían como parámetro y lo almacenamos en una variable **mensaje**, esta variable será la que se enviará al servidor.

```
let mensaje="";
for (let j = 2; j < process.argv.length; j++) {
  mensaje+=(process.argv[j]+" ");
}
```

Finalmente al momento de realizar la conexión enviamos el mensaje que recibimos como parámetro.

```
var net = require('net');

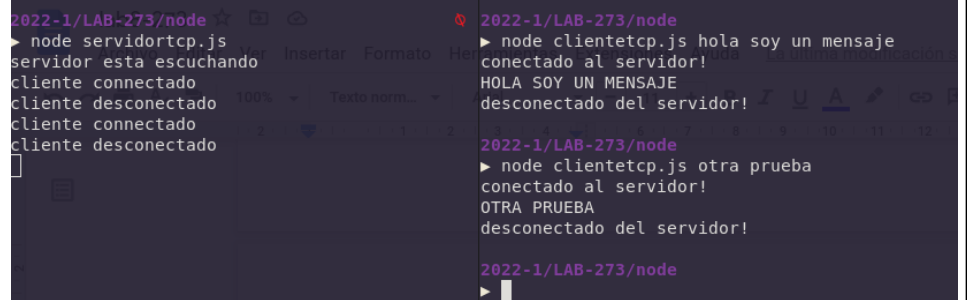
let mensaje="";
for (let j = 2; j < process.argv.length; j++) {
    mensaje+=(process.argv[j]+" ");
}

var client = net.connect({port: 8080}, function() {
    console.log('conectado al servidor!');
    client.write(mensaje);
});

client.on('data', function(data) {
    console.log(data.toString());
    client.end();
});

client.on('end', function() {
    console.log('desconectado del servidor!');
});
```

Comprobamos el funcionamiento de ambos programas



```
2022-1/LAB-273/node
> node servidortcp.js
servidor esta escuchando
cliente conectado
cliente desconectado
cliente conectado
cliente desconectado

2022-1/LAB-273/node
> node clientetcp.js hola soy un mensaje
conectado al servidor!
HOLA SOY UN MENSAJE
desconectado del servidor!

2022-1/LAB-273/node
> node clientetcp.js otra prueba
conectado al servidor!
OTRA PRUEBA
desconectado del servidor!

2022-1/LAB-273/node
>
```

2. Desarrolle en Node.JS un cliente HTTP que se conecte a un servidor Web, envíe una petición GET y descargue el archivo solicitado mostrando la cantidad de bytes de la respuesta recibida del servidor web. Su programa deberá usar los módulos `https` y `fs`.

importamos las librerías necesarias	<pre>var http = require('http'); var fs = require('fs'); const url = require('url');</pre>
definimos la url a donde se conectara y se hace un tratamiento para obtener el nombre del archivo a descargar	<pre>var enlace='http://www.fcpn.edu.bo/wp-content/uploads/2021/12/instituto_ie.jpg' var aux=(url.parse(enlace).pathname).split('/') var namefile=aux[aux.length-1];</pre>
se hace la creacion de el archivo a descargar y almacenar	<pre>var file = fs.createWriteStream(namefile); var request = http.get(enlace, function(response) { response.pipe(file); });</pre>
codigo completo	<pre>var http = require('http'); var fs = require('fs'); const url = require('url'); var enlace='http://www.fcpn.edu.bo/wp-content/uploads/2021/12/instituto_ie.jpg' var aux=(url.parse(enlace).pathname).split('/') var namefile=aux[aux.length-1]; var file = fs.createWriteStream(namefile); var request = http.get(enlace, function(response) { response.pipe(file); });</pre>

comprobamos
funcionamiento

el

