What is network, client-server architure, basic protocols and their similarities and differences?

**Network** is a connection between client and server.

**How data flows** between network is like:

1. Client requests data from server

2. Load balancer routes the request to the appropriate server

3. Server processes the request client

4. Server queries appropriate database for some data

5. Database returns the queried data back to the server

6. The server processes the data and sends the data back to the client 7.

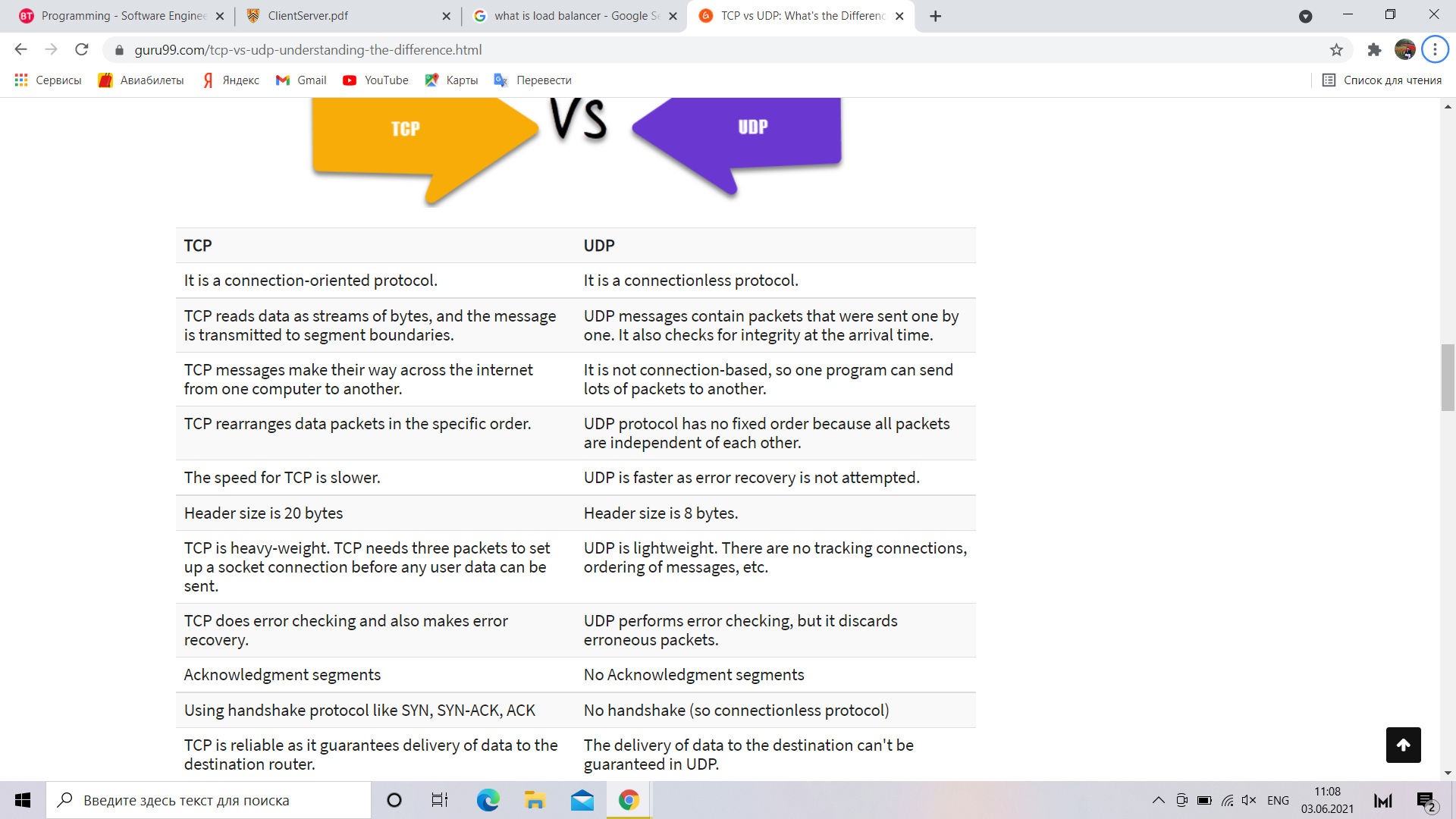
This process repeats, although there are also some different type of architecture but this one is the most typical.

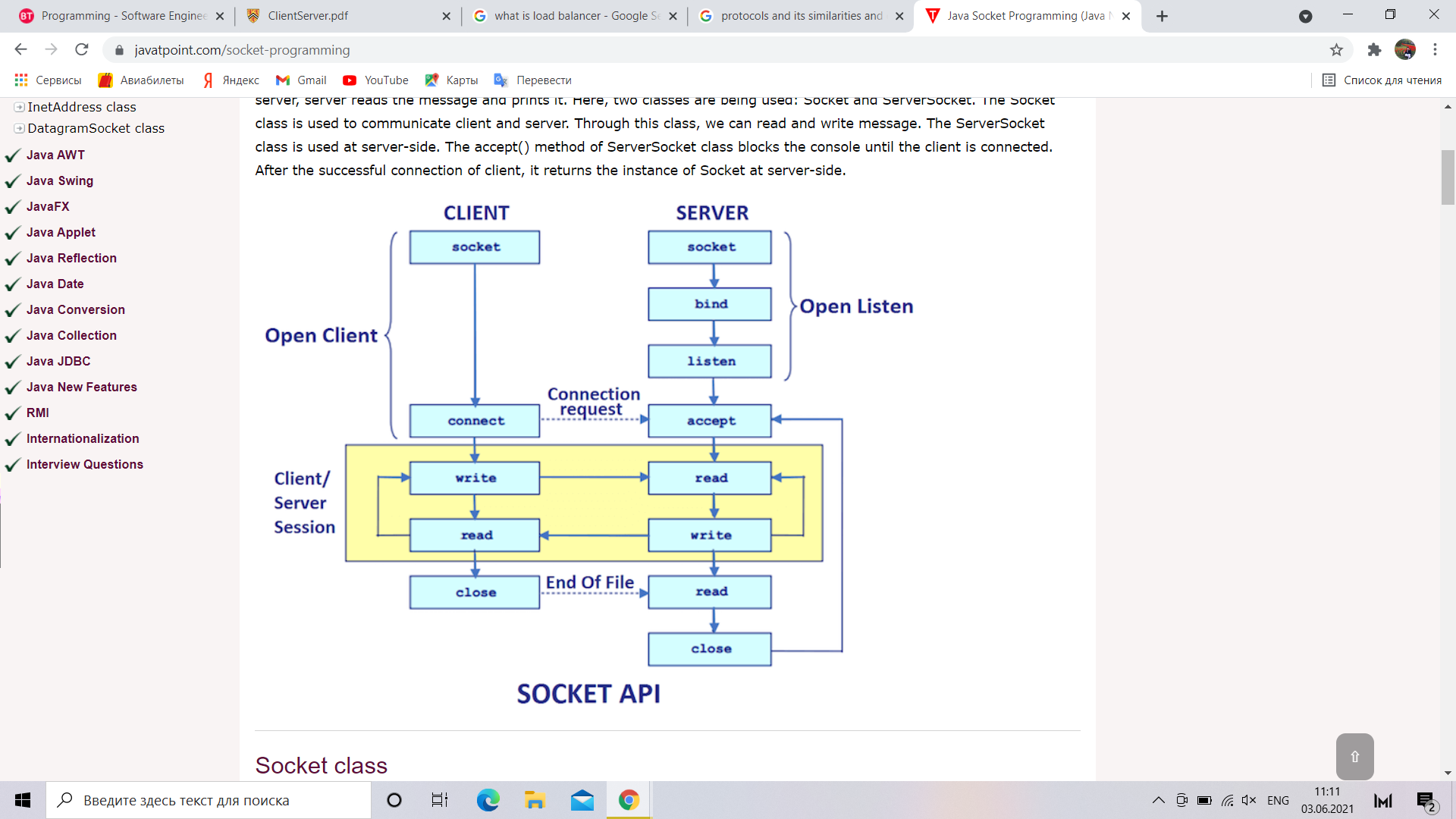
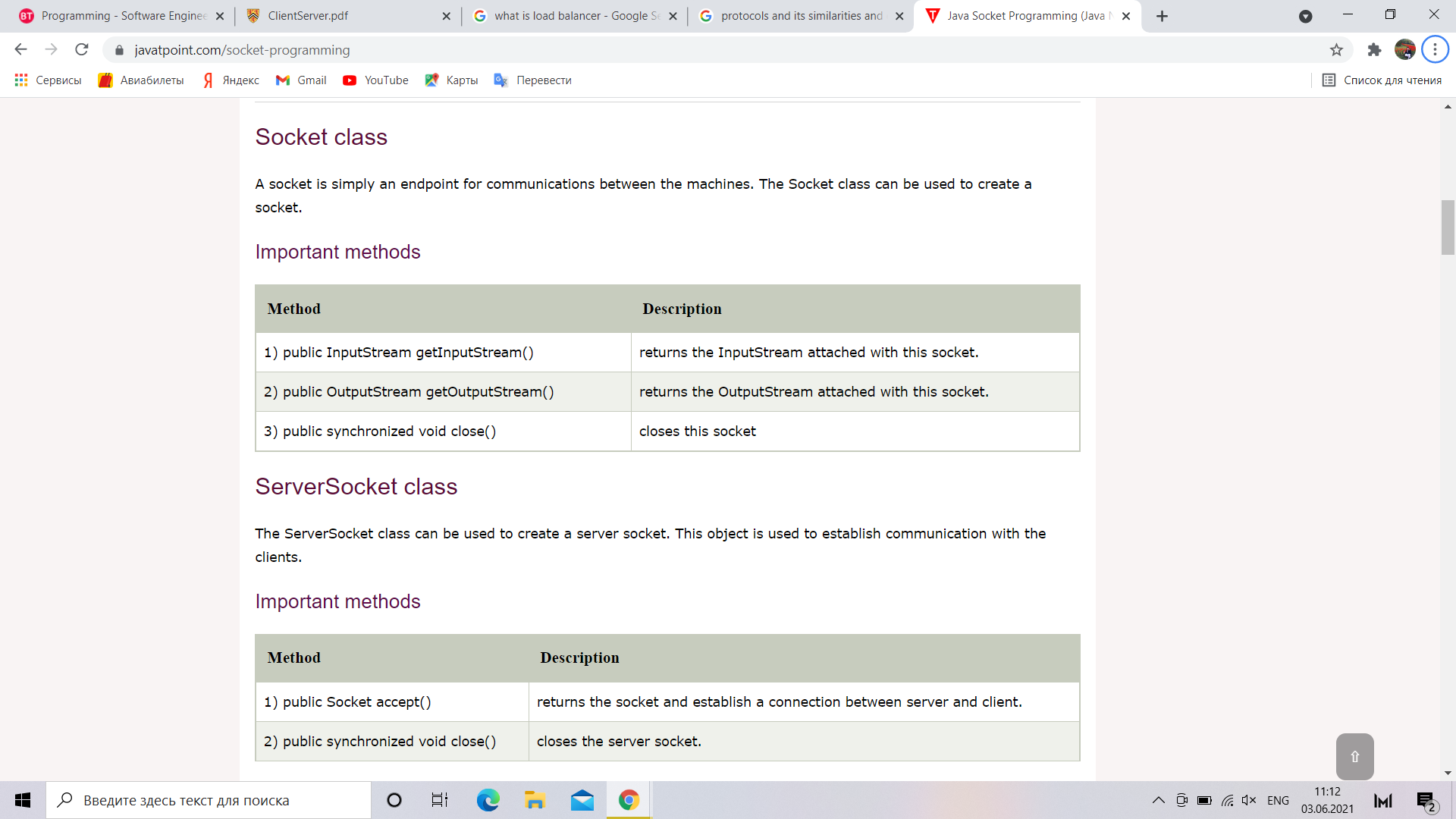
**Protocols** help network not only make devices connected to the network but also give them ability to interact with data or requests.

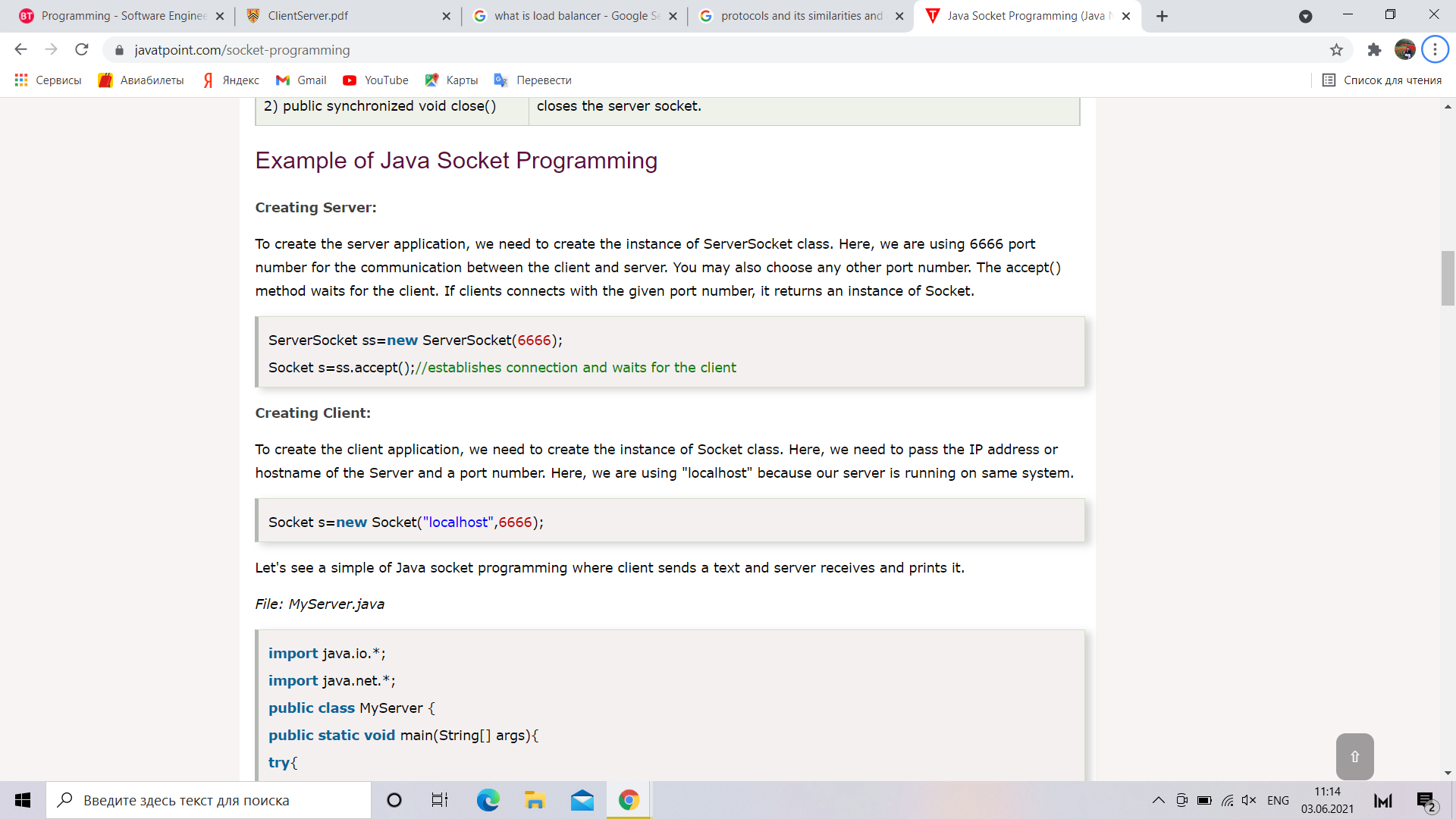
Most famous protocols are TCP and UDP.

Both of them are related to the Transport layer of OSI model, which basically use in transfer of data but with different manners.

**TCP**/IP stands for Transmission Control Protocol/ Internet Protocol. It is specifically designed as a model to offer highly reliable and end-to-end byte stream over an unreliable internetwork.

**UDP** is a Datagram oriented protocol. It is used for broadcast and multicast type of network transmission. The full form of UDP is User Datagram Protocol (A datagram is a transfer unit associated with a packet-switched network.) The UDP protocol works almost similar to TCP, but it throws all the error-checking stuff out, all the back-and-forth communication and deliverability.





A **ServerSocket** is for accepting incoming network connections on some stream protocol; e.g. TCP/IP.

A **DatagramSocket** is for sending and receiving datagrams on some connectionless datagram / message protocol; e.g. UDP/IP

**Java DatagramSocket class**

Java DatagramSocket class represents a connection-less socket for sending and receiving datagram packets.

A datagram is basically an information but there is no guarantee of its content, arrival or arrival time.

**Commonly used Constructors of DatagramSocket class**

*DatagramSocket*() throws SocketEeption: it creates a datagram socket and binds it with the available Port Number on the localhost machine.

*DatagramSocket*(*int port*) throws SocketEeption: it creates a datagram socket and binds it with the given Port Number.

*DatagramSocket*(*int port, InetAddress address*) throws SocketEeption: it creates a datagram socket and binds it with the specified port number and host address.

**Java DatagramPacket class**

Java DatagramPacket is a message that can be sent or received. If you send multiple packet, it may arrive in any order. Additionally, packet delivery is not guaranteed.

**Commonly used Constructors of DatagramPacket class**

DatagramPacket(byte[] barr, int length): it creates a datagram packet. This constructor is used to receive the packets.

DatagramPacket(byte[] barr, int length, InetAddress address, int port): it creates a datagram packet. This constructor is used to send the packets.

**ServerSocket vs DatagramSocket:**

A **ServerSocket** is for accepting incoming network connections on some stream protocol; e.g. TCP/IP.

A **DatagramSocket** is for sending and receiving datagrams on some connectionless datagram / message protocol; e.g. UDP/IP

**Design Patterns** in Java

A design patterns are well-proved solution for solving the specific problem/task.

Advantage of design pattern:

They are reusable in multiple projects.

They provide the solutions that help to define the system architecture.

They capture the software engineering experiences.

They provide transparency to the design of an application.

They are well-proved and testified solutions since they have been built upon the knowledge and experience of expert software developers.