FUNDAMENTALS OF OBJECT ORIENTED PROGRAMMING

Lecture54

Lecture 54: Communication over HTTP and Related Protocols

Welcome to lecture 54, Advanced Topics in Object Oriented Programming. So, in this lecture, I will start with HTTP and REST basics, alright. So, we know Hypertext Transfer Protocol, right, HTTP. So, usually you use in the browser, right. So, whenever you want to go to some website, most of you would have used this.

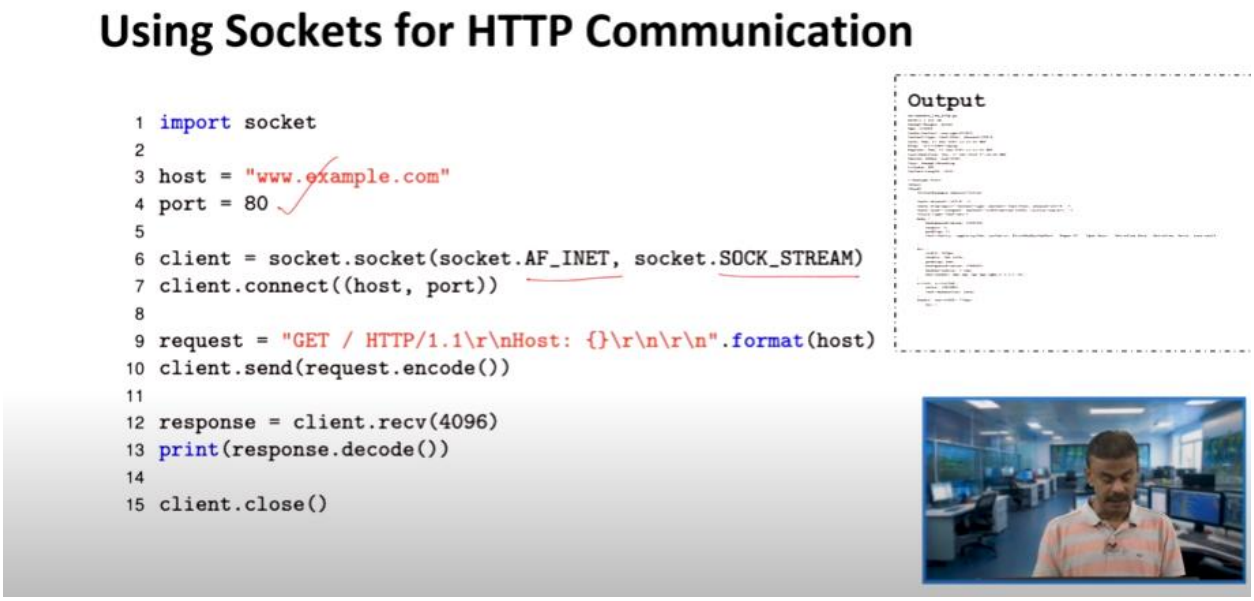
So, this is a stateless protocol for transferring data over the web. For example, if you want to download or if you want to upload, right, or you want to see certain information, right. So, that means that the data is being transferred. So, it is transferred through the web. So, there are certain commands or certain common methods like a get, post, put and delete.

So, we will see. So, the next one is a representational state transfer. right, we call it as REST. This is nothing but architectural style for designing networked applications, right. So, this uses standard HTTP methods for CRUD.

So, that stands for create, read, update and delete operations, right. So, we will see, suppose we use, let us say, get slash user slash one. So, this will retrieve user details, right. So, get command or get method will retrieve user details and another one is the post post slash user users right

so this will add a new user right so this is a basics of this http and the best so we will see with an example in python right so we are going to see with the help of sockets for right or using sockets for http communication So let us import socket. All right. So let us import socket.

And the host here is www.some website example.com. Right. So you can change whatever the website that you want to put. You can do that. So this is the host name.



And or you call it as a domain name. Host name or a domain name. All right. So where we are connecting to. Port is equal to 80.

Line number 4. Port is equal to 80. So this is specifying. the port number right so this is specifying the port number so port 80 is the standard port for http traffic right port 80 it is

nothing but the standard port for http traffic so next line you have client is equal to the socket is invoking socket of

so we know af inet is for ipv4 all right and the sock underscore stream this is specifying the TCP protocol for reliable connection oriented communication right. So, this we have already seen right. So, one is for IPv4 and another one is the TCP protocol for reliable connection oriented communication. So, next line you have client which is invoking connect right by passing host and port right.

So, this is establishing a connection to the server right. So, this is establishing a connection to the server at the specified host right for example here we have www.example.com right and port 80 so host is www.example.com this is your passing and the port is 80 so this is up to line number seven and line number nine you have request is equal to right so you are having some statements get slash

HTTP slash 1.1, then backslash or and backslash n host, right. So, then there you are having certain command, we will see. And then with this dot, okay, the format is invoking host, right. The format is being invoked where host is being passed, right. Format is being invoked, host is being passed.

So, now if you look at line number 9, so you have get, right, get slash HTTP, right, the meaning is it prepares an HTTP get request to retrieve the root resource, right, you are having slash, right, so this slash, right, so slash of web server, right. So, HTTP get, right, request, right, HTTP get request to retrieve the root Resource slash.

All right. So this is a slash of the web server. So now you have get slash. Right. So this asks the server to retrieve the root resource.

And then you have HTTP slash 1.1. Right. So this specifies the HTTP version being used. That means 1.1. Right.

Is the most common. 1.1 is the most common, so that is what we are using right now: 1.1. And then you have host, right? So host, host with this host colon—this one I am considering. So this is required in HTTP slash 1.1 to specify the domain name of the server. So in this case, you have what? www.example.com.

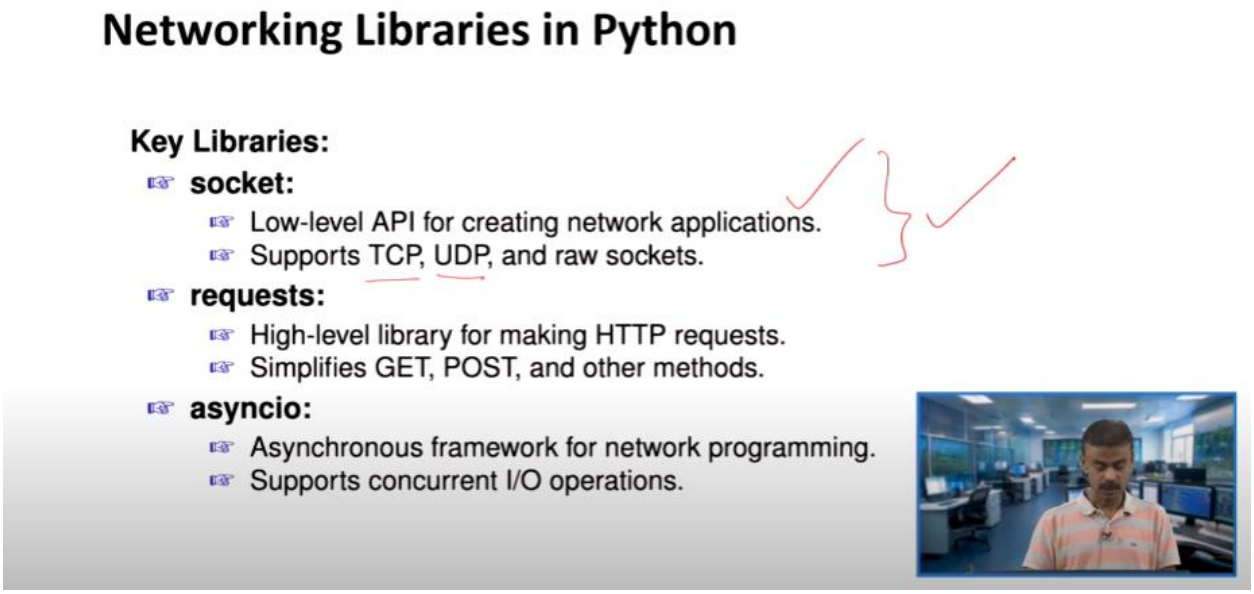
So www.example.com. And then you have sequence of slash R slash N. So this is nothing but the carriage return and new line. Slash R is for carriage return. Here also you are using, right? And slash N is for the new line, right?

So which are required to format the HTTP request properly. So here you have slash r slash n slash r slash n. So which is nothing but the end of the HTTP request address. So this is all about your line number 9. Next what is happening? The client is invoking send.

You are passing request and encode. You are passing request and encode. So this is sending the HTTP request to the server. So this is sending HTTP request to the server request dot encode. So this you are passing the request is invoking encode function.

So this is converting the string into bytes. So, this is converting string into bytes. So, this is required for sending over a socket. So, up to line number 10 is over now go to line number 12. So, line number 12 client is invoking the received with 4096.

So, what is the meaning? So, this is nothing but it is receiving the servers response right receives the server response. So, 4096 which is specifying The maximum number of bytes, right, the maximum number of bytes to read from the server at once, right, to read from the server at once, so 4096 bytes, right, it is receiving the server's response and the number 4096, it is specifying the maximum number of bytes, max number of bytes, right, to read from the server at once, right.



So, up to line number 12 now we have completed and if you go to line number 13 the print statement right. So, here you have the print right which is a function response is invoking decode right. So, that means it is decoding the received bytes into a string right

and prints the servers response to the console right. So, the response dot decode. So, when it is invoking

So this is decoding the received bytes, right? Decoding the received bytes. So that is converting into a string, right? And prints the server's response to the console, right? So the response will typically be the HTTP status line, right?

So which is nothing but HTTP slash 1.1 and then 200, okay, right? So something like that will be printed. So, that is a status line in fact. So, that is nothing but a status line. And the response headers.

The response is nothing but HTTP status line 1. And the second one is response headers. So, response headers are nothing but content length or even both. So, next one is the actual HTML content of the web page. So, the third one is the actual HTML content of the web page.

So, what I suggest here you take some other examples. All right. In the www.example.com, instead of that, you take some website. All right. And then you can try this.

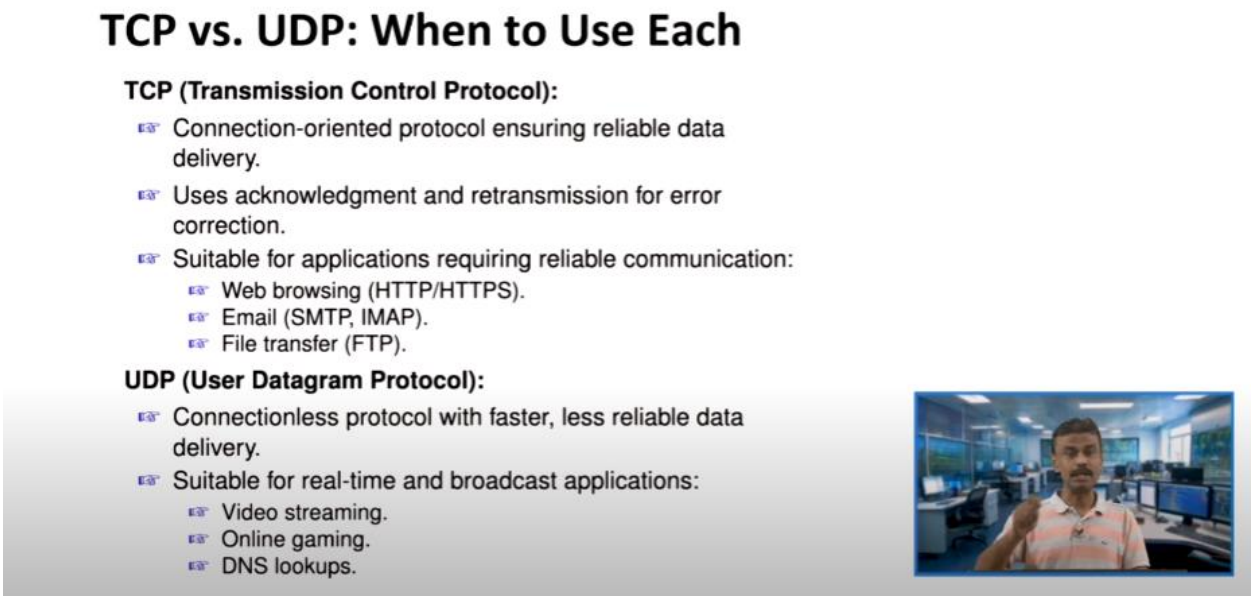
So finally, you have client close. So that means so this is closest the socket connection to the server. All right. Freeing up the resources. So now when you run the code, you will get the output like this.

So, this is some dummy www.example.com. So, that is why it is not been projected right purposefully I have projected. So, what you can do what I suggest you take some website right and see the code right understand and when you run it you will get the output like this. So, maybe if you run the code. So, let us say in the python we will run the code you will understand yeah.

So, in the python so, let us say python 3. So, you can see that right. So, some example domain all right and then we are getting this is the sample output that I put. So, what I suggest you take some website all right. So, consider some website as the input all right.

So, you are writing all right. So, what we have done? So, we have considered the host right www.example.com. So, in a similar way you consider some other website and so you can see the complete output like this. okay so i hope it is clear now so next one is a

networking libraries in python so these are all the key libraries so when i am talking about socket



so it is low level you have application programming interface api so it's a low level api for creating network applications socket is a low level api and it supports tcp the transmission control protocol as well as user datagram protocol right and raw sockets all right. So, this supports TCP UDP and raw sockets. So, this is one of the key libraries.

The second library is a request all right. So, here you have high level library for making HTTP requests all right. So, requests you have the high level library for making HTTP requests and it simplifies the methods like a GET POST etcetera. And another library is asynchronization one you call it as a asyncio. So, for the asynchronous framework for this we had seen case of C++ and Java right.

So, we can use this library right for this particular framework the asynchronous framework right when you are using the network programming. So, you have the library called asyncio and it is supporting concurrent input output operations right. So, when you have the concurrent input output operations. So, we can use the asyncio function or you call it as a library so now what is the difference between the tcp and udp right just now we talked in the case of python library right so when you have to use both of them right

so as we have seen tcp is nothing but the transmission control protocol all right it is nothing but the connection oriented protocol so ensuring you have a reliable data delivery right so you are getting a reliable data so either you are sending or receiving right so your data should be the reliable one so the tcp is nothing but the connection oriented protocol and this tcp uses acknowledgement and retransmission for error correction all right and for web browsing like http https emails right smtp server or imap server file transfer right you call it as a ftp all right so these are all the suitable applications for tcp so whenever all right

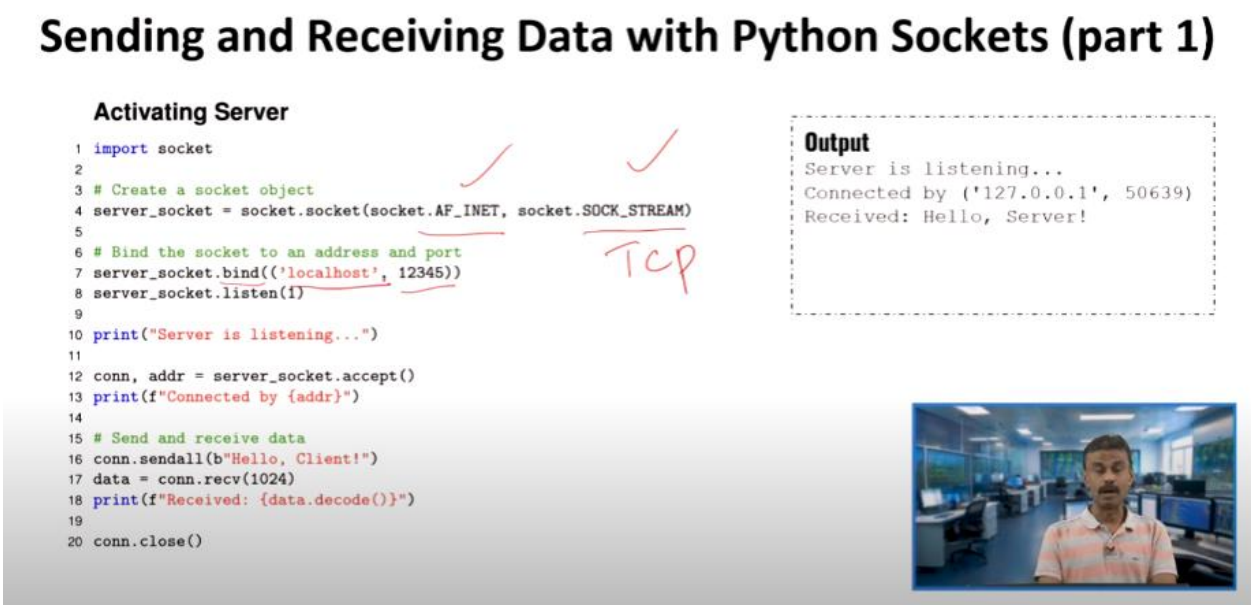
You are doing web browsing, which is based on the Transmission Control Protocol. Even sending an email or receiving an email, all right? SMTP server—you might have heard of SMTP server, IMAP server, right? So, the email servers use TCP. And similarly, File Transfer Protocol, FTP, right? So, these all come under the Transmission Control Protocol. The next one is User Datagram Protocol, right?

Video streaming. So, nowadays, this is very popular, right? Or online gaming, right? So, many of you play online games. So, that is based on the User Datagram Protocol, right?

Similarly, DNS lookups, right? So, this is the connectionless protocol where it is very fast but less reliable for data delivery. Right? So, compared to TCP, this is less reliable for data delivery, right? Nowadays, we are very familiar with all the online meetings, right? So, we need video streaming. Suppose you want to present something—we need video streaming. So, that means UDP has to come to your mind, right? So, that means, with the help of the User Datagram Protocol, we can do online meetings.

you can share your videos right similarly online gaming very popular another one It is based on the UDP, the user datagram protocol, alright. So, these are all some of the cases or applications where you use TCP, right. Web browsing, email or file transfer, we use TCP. Whereas, the online gaming or video streaming, we use UDP, alright.

So, this is the difference between TCP and UDP. So, one is the connection oriented protocol and it is ensuring the reliable data delivery. So, whereas another one is connection less protocol with faster but less reliable data delivery right. So, this is a difference between TCP and UDP. So, next what we can do we will see how you can send and receive data with python sockets right.



So, import socket. So, this we have done and the next line number 4 you have the socket which is invoking by passing AF underscore INET. So, this is we know this is for IPv4 and the SOCK underscore stream this also we had seen. So, this is specifying the TCP protocol for reliable connection oriented communication just now we had seen right.

So, these two are for that right. So, the next line the server underscore socket which is invoking the bind function right. So, this we had seen in C plus plus and Java if you recall. You are pausing local host and some number, let us say 1, 2, 3, 4, 5, right? So, the local host, nothing but the server will listen for connections on the local machine, right?

That is a meaning, local host, right? The server will listen for connections on the local host or on the local machine, right? 1, 2, 3, 4, 5, you have a number, right? This is the fourth number. Right.

So this is a port number on which the server will accept connections. The server will accept connections. And in the next line server underscore socket which is invoking listen and you are passing 1. Right. So this is preparing the socket to accept incoming connection requests.

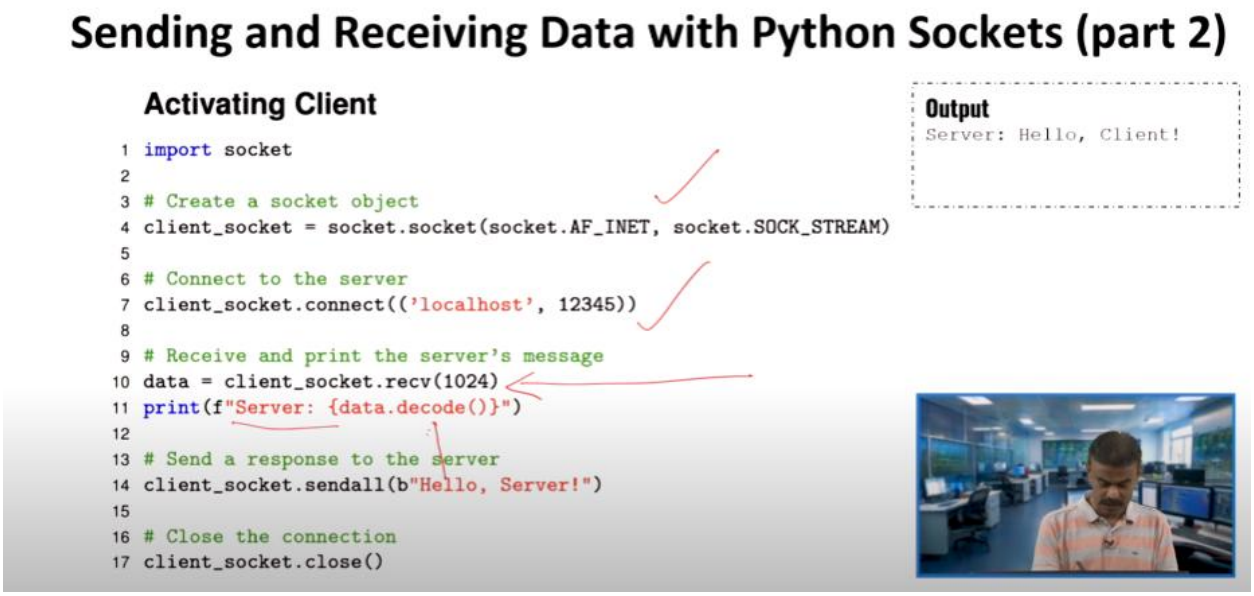
Right. Incoming connection requests. So, you are specifying The maximum number of pending connections in the queue. Right.

You are specifying one here. We are specifying one here. So, the meaning of one is. Right. We are specifying the maximum number of pending connections in the queue.

Right. So, up to line number eight, we completed. So, the next line, line number 10, you have print. Right. Server is listening.

Right. So this is printing a message. right, indicating that the server is ready and waiting for client connections, right. So, that is the meaning. And then you have, right, the connection comma address that is equal to server socket is invoking the accept function, right.

So, here you have the connection and address c o n n, right. So, mainly the full meaning of this line number 12, which is nothing but this is waiting for an incoming client connection. and CONN is for a new socket object right CONN is the new socket object for communicating with the connected client and then you have address ADDR. So, which is nothing but address IP and port all right. So, this is IP and



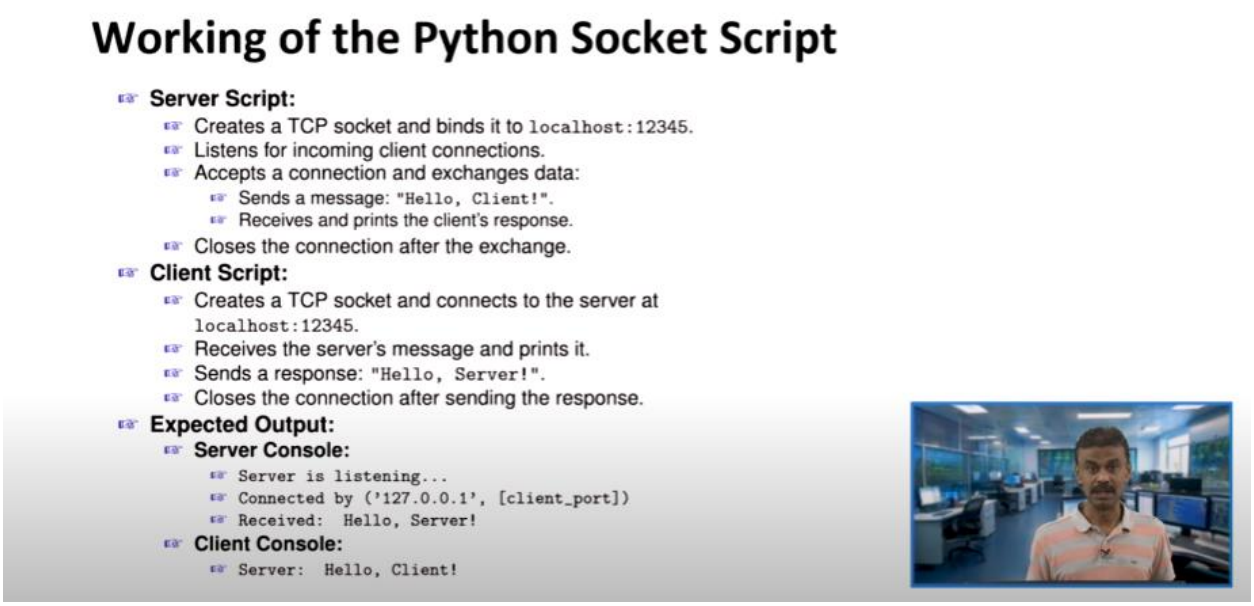
port of the connected client of the connected client all right so then the server will block until a client connects all right so block means it is passing the execution correct so that is the meaning of the line number 12 and then you have a print statement so it is printing the address of the client that connected to the server line number 13 prints right address of the client that connected to the server. So, the object CONN which is invoking send all right you are passing where we have that hello client right.

So, the meaning is it is sending a message to the connected client right it is sending the message to the connected client. Here you have B and then you have double quote hello client right. So, that is what you are having over here So, this is the message which is encoded as bytes before being sent, right. So, this is encoded as bytes before being sent.

And then next line you have the object CONN which is invoking receive by passing your passing 1024, right. So, this is receiving data from the client, right. So, this is receiving data from the client, right. So, 1024 which is nothing but the maximum number of bytes to read at a time right 1024 the maximum number of bytes to read at a time all right.

So, now the function blocks all right. So, this is passing the execution until the data is received all right it is passing the execution until the data is received. So, line number 17 done. So, now you have print statement. So received data is invoking decode.

So the meaning is it is decoding the received bytes into a string. Exactly we had seen before. Received bytes into a string and prints the message received from the client. So then connection is invoking close. Once you execute this particular program, we are going to get the output like this.



Server is listening, connected by, alright. So, now some numbers we are getting and then you are having received allow server, ok. So, what we can do? We will run the code using Python, right. So, let us see.

we will go to the python now we are activating server we are in activating server slide so here i will run python 3 right so let us see now you can see we have got the first line as the output right server is listening all right so now we have activated server So, next we are going to see right activating client. So, once we activate client and then run the code the next two lines will be printed right. So, that what we can do I will continue with activating client.

So, then we will see the output. So, let it be there as it is all right. So, now what we will do we will go to the next slide and we will see how you are going to activate the client. So, let us see when you activate client. So, this is the

So, import socket rest of them are same this also we had seen and client socket local host and 1, 2, 3, 4, 5 this also you know that all right. So, we have explained several times and line number 10 right. So, the client socket is invoking receive 1024. So, we have already seen this line is nothing but receiving a data from the server right. So, we know that the 1024 which is the maximum number of bytes to read at a time right.

So, now the function blocks until data is received or the server receives. closes the connection so the next line you have print server right so data is invoking decode so that means the decodes it is nothing but decodes the received bytes into a string and prints the server servers message and it is printing the servers message so now you have the client socket which is invoking send all right so here you have allow server Right.

So it is in the printed, I mean, code. So that means it is sending a response message to the server. Right. It is sending a response to the server. So the message, this particular message is encoded as bytes before being sent.

Right. And then send all. So this is ensuring that all bytes to the message are sent. Right. So then you have the client socket, which is calling the close.

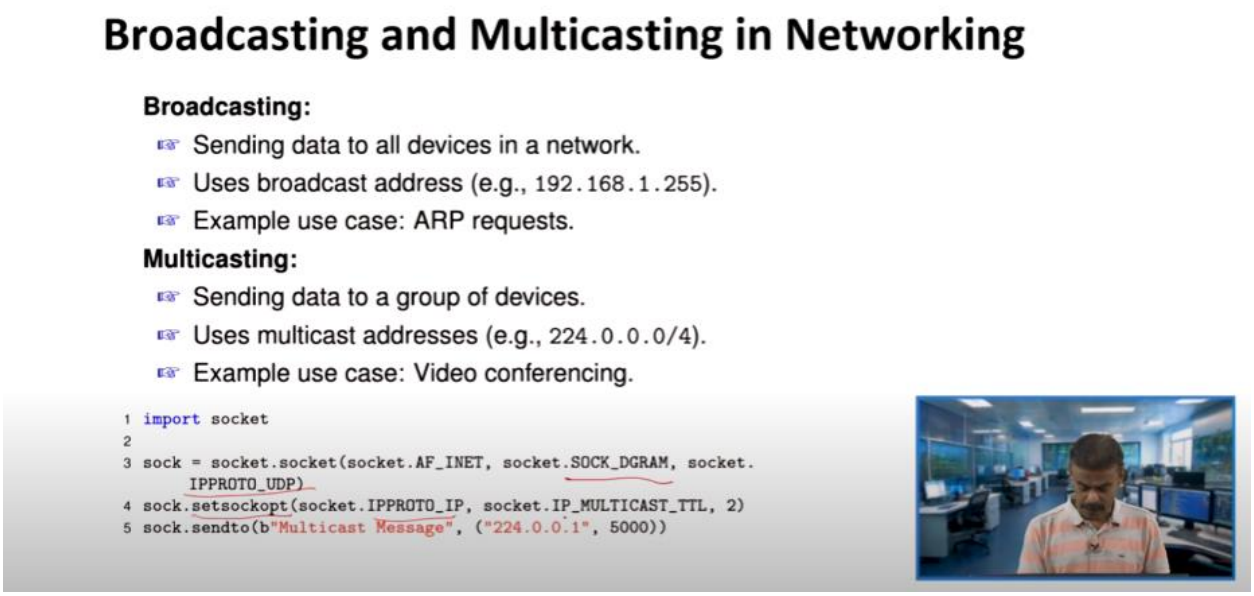
So, when we run the code, so we will go to the python because there is a connection between part 1 and part 2. So, we will see right. So, we will run the code. So, here we are getting server hello client and now you can see the previous program right. So, you can see the output right.

So, maybe we can run one more time python 3 right. In fact, we are seeing the output here, but for the confirmation. We re ok server is listening. So, we will run this one again. Once you are running and it is server is saying hello client and then here you can see right the connected by 127.001 and then you are having the number right IP and port and then it is saying received hello server right.

So, previously when we run we have got only the first line server is listening right and dot dot dot right. So, these things what you can do you can try. So, these are the two outputs we had seen and we are able to print those two output. So, whatever we have done right the previous script that is you call it as a Python socket script. So, the server client we have done right.

So, we have created a TCP socket and binds it to the local host 1, 2, 3, 4, 5. So, that is what we have done in line number 7 and it is listening for incoming client connections right. So, here you have the listen of 1 all right. So, all these I explained right. So, for your reference I have put as a text.

So, the server script is accepting a connection and exchanging the data. In fact, it is sending the message, 'Hello client,' right? So, those things we have seen. First, we saw the output, and then it is closing the connection after the exchange. So, in fact, it is receiving and printing the client's response.



So, when you go to the client script, because only one line was seen as output, all right? So, once we run the client script, right? So, in fact, in the case of the client script, it is creating a TCP socket and connecting to the server at, the localhost, right? Some number, and it is receiving the server's message and printing it, right? And it is sending a response to the server. And it is closing the connection after sending the response. So, in fact, we have tested both and looked at the server console

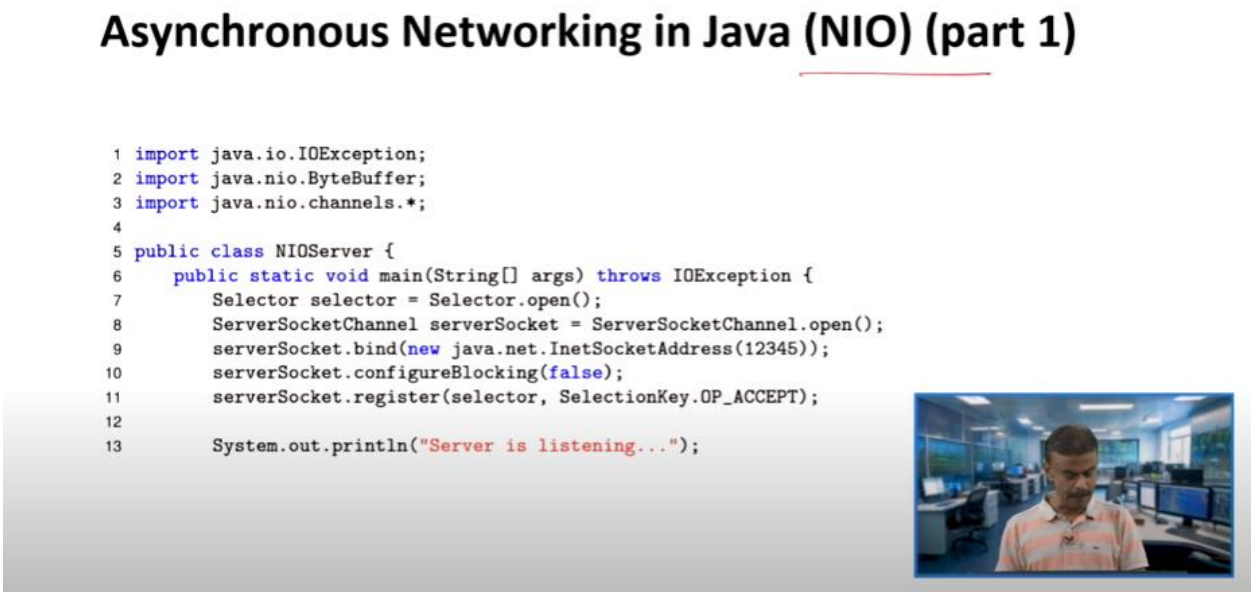
and client console. We have got the said output, right? So, this output we have seen in both the server console as well as the client console. I hope it is clear now. So now we have the concept of broadcasting. What do you mean by broadcasting and multicasting in networking? Right.

So broadcasting. So when it is sending data to all devices in a network. Right. Assume that you want to send, let us say, a particular message or email. right so to all the devices in the network for example you have address resolution protocol requests right arp requests

you call it as address resolution protocol requests right so the broadcasting which is using the broadcast address some address so for example here 192.168.1.255 right so this is using the broadcast address ARP request is an example. Address resolution protocol. So this is broadcasting.

That means you are sending the information or a data to all the devices in a network. A multicasting. What do you mean by multicasting? So sending a data to a group of devices. You are sending to the group of devices.

For example, video conferencing. So video conferencing. Nowadays after COVID, we are very familiar with this video conference. So many meetings, so many conferences, online conferences. So that is an example of a multicasting.



So, this is using the multicast addresses like the one you are seeing over here 224.0.0.0 slash 4 right. So, this is a multicast address this is an example of a multicast address. So, now here I have written a snippet right import socket and the socket is invoking socket of right AF underscore INET. So, that you know it is a IPv4 addressing and the socket datagram. So, this is indicating this is a UDP.

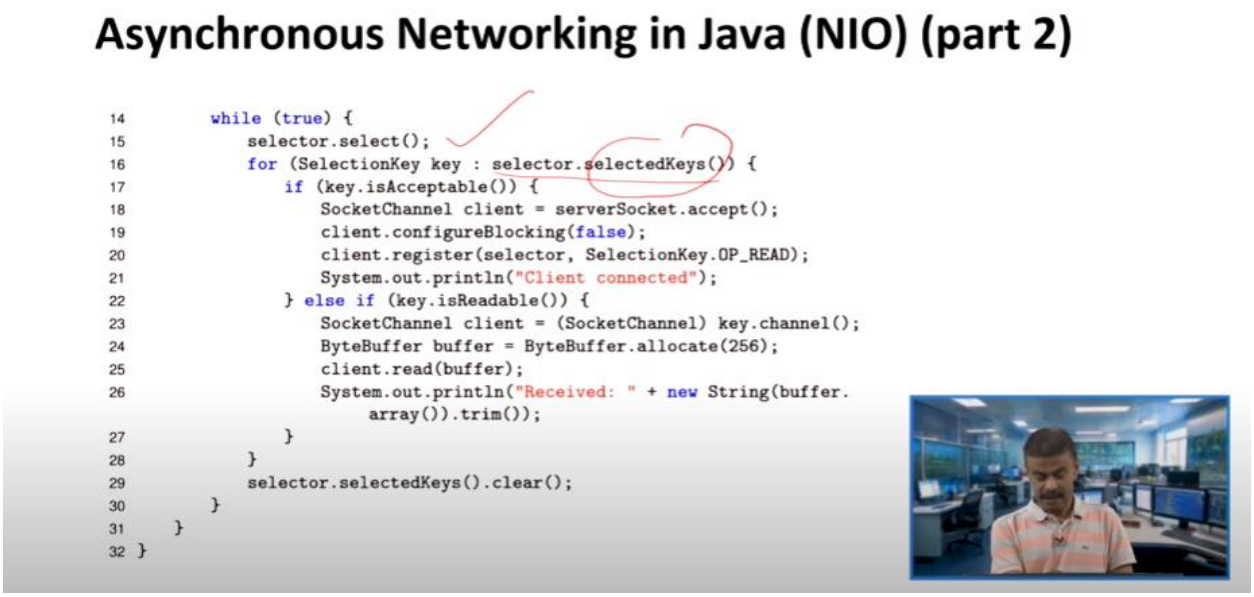
This is a UDP datagram socket. This is a UDP datagram socket. And the next one IPPROTO underscore UDP which is nothing but the UDP protocol will be used. So, this is specifying the UDP protocol will be used. And the next line SOC is invoking sets SOC option and socket dot IPPROTO underscore IP.

And socket dot IP multicast underscore TTL and then 2. So, what is the meaning of this? IP photo underscore IP, right. So, which is specifying that the option applies to the IP level. Specifying the option applies to the IP level, right.

So, next one is IP underscore multicast underscore TTL. So this is setting time to live, right? TTL stands for time to live. Time to live, right? So TTL stands for time to live for multicast packet, right?

So this TTL, which is controlling the number of hops a packet can travel before it is discarded. So the packet can travel before it is discarded. And setting it to 2, you are seeing 2. So that is allowing the packet to travel across up to 2 routers. The packet to travel across up to 2 routers.

So that is the meaning of this particular line. So now this SOC is invoking send to. You have multicast message and you have port and number. That means IP address and the port, right? IP address and the port.



So here you have multicast message, the data to be sent encoded as bytes, right? And 224.001 is a reserved multicast IP address, right? The reserved multicast IP address. So this IP address belong to the range 224.0.0.02, right? 239.255.255.255 right.

So, starting from 224.0.0.0 to this right. So, which is reserved for multi-car communication and 5000 right the destination port number. So, this is nothing, but destination support number right. So, this is a snippet of python.

So, the next concept asynchronous networking in java right. So, again NIO right. So, we are coming back again to java. So, NIO stands for new input output right NIO stands for new input output. So, we will see two code.

So, that is why I put part one here. So, let us see. So, we are importing java IO exception byte buffer channels dot star. So, these are all NIO dot byte buffer new input output IO you know input output here N stands for new and third line you have Java nio channels dot star right.

So, you have a public class which is a driver class NIOServer and here you have the main program starts throws IO exception. So, now line number 7 you have an object

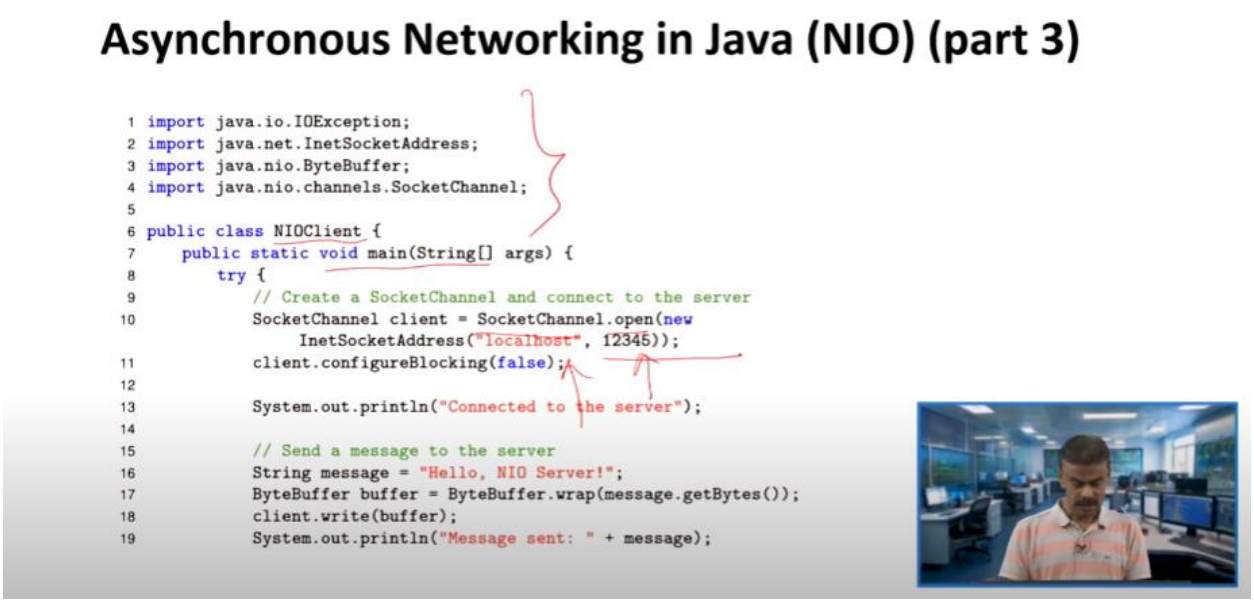
selector under selector right. So, which is invoking open right. So, here you are seeing selector right.

So, which is selector this one small s right. So, which is a multiplexing mechanism. So, this is allowing a single thread to monitor multiple channels for events like read or accept right. So, for read or accept this is a multiplexing selector is the multiplexing mechanism. So, line number 8, you have server socket under server socket channel because you have a new input output.

So, you have all these classes and under this, you have an object server socket, right? So, which is again invoking the open, right? Which is line number 8 and line number 9, you have the bind method. right so the bind method which you are pausing the inet socket address right so the socket address is one two three four five line number nine and then you have server socket which is invoking configure block false right

so here you have line number nine inet socket address which is one two three four five right so the server socket channel all right so under server socket channel so which is being created Which is being created that means server socket object is being created and it is bound to port 1, 2, 3, 4, 5 that is a meaning right and it is bound to port 1, 2, 3, 4, 5 and then you have one more right server socket which is invoking configure blocking false right. So, which is nothing but the non-blocking mode which is enabling with server socket which is invoking configure blocking false.

Right. So up to line number 10, we have done. And then line number 11, you have the register. Right. That method by pausing selector selection key dot OP underscore accept.



Right. So the meaning is the OP underscore accept event. Right. So this is an event which is indicating that the server is ready to accept new connections. Right.

The server is ready to accept new connections. So selector already we know. all right, this object we already we know. So, when it is invoking a register method, where the class is, all right, in fact, the server socket, the object server socket, when it is invoking the register method, you are passing selector and the selection key dot op underscore accept. So, this is nothing but an event, all right.

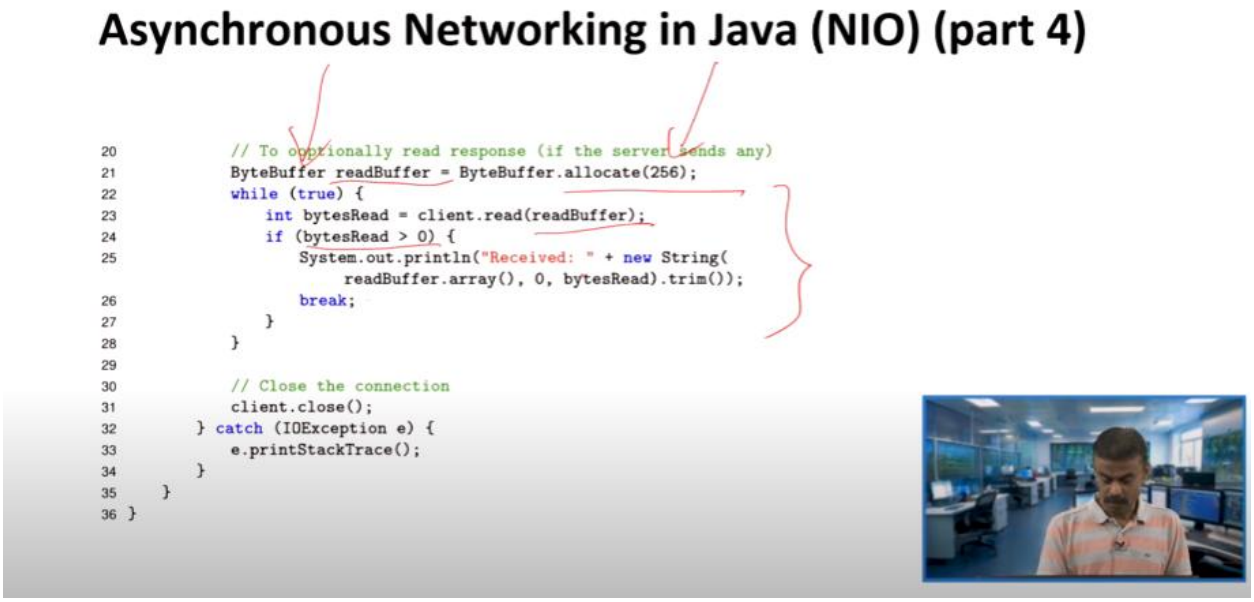
So, this indicates that The server is ready to accept new connections. Okay. So up to line 11, we have completed. So line 13, you have System.out.println.

Server is listening. That means this is a simple print statement. So this indicates that the server has been initialized and is ready to handle client requests. Okay. So now, while true, selector.select.

So this method, selector.select, selector is invoking select. So that means this method blocks until at least one of the registered channels is ready for input-output operations. For example, you have accept, read, or write, right? Accept, read, write, correct? So I'm repeating again.

So when it is invoking the selector, which is invoking select, right? So this method blocks until at least one of the registered channels is ready for input output operations for example accept read write right. So, the next one you have selected keys right the selector dot selected keys. So, range for loop the selector is invoking selected keys that

means this is returning the set of selection keys representing the channels that are ready.



representing the channels that are ready so it will be assigned to key right so that range for loop we know you have now if statement if key is invoking is acceptable that is that means if it is true that is it is checking if a new client connection is ready to be accepted right then you have following statements right otherwise I will explain about that, what are all the statements. Otherwise, I will explain these four lines. So, if key is invoking is readable, right, what is the meaning of this?

It is checking if a connected client has data to be read, right. It is checking key dot is readable, key is invoking is readable. So, it is checking if the connected client has data to be read. So, now suppose that key is invoking is acceptable assume that if this is true right. So, then you can see that you have an object client under socket channel right and the right hand side the server socket is invoking accept right.

So, the meaning is it accepts a new client connection right it is accepting the new client connection and in line number 19 you have Client is invoking configure blocking by passing false, right? The meaning is this configures the client socket channel to non-blocking mode, right? The meaning is, so this is configuring the client socket channel to non-blocking mode. And then again, line number 20, client is invoking register.

We are passing selector selection key dot op underscore read. right the meaning is so this is registering right we are at line number 20 so this is registering the client channel with the selector for read operations and then you have system dot out dot parental and client connected right if is acceptable is true suppose key is invoking is readable is true right line number 17 is false obviously it will go to the else part

right and here you have a client object under socket channel so key is invoking channel and buffer is an object under byte buffer so byte buffer which is allocating the maximum 256 bytes for incoming data right and then you have client is invoking read by passing buffer right so that means it is reading a data from the client channel into the buffer right that is the meaning reading data from the client channel into the buffer and then you have system dot out dot printing.

So, here you have a received right the string which is invoking right the string. So, buffer is invoking array and then you have trim correct. So, that means string buffer dot array and then trim that means prints the it is nothing, but it is printing the received data into the console right. So, you can look what are the trim function is doing buffer dot array is doing right so that means it is going to print the received data into the console right and then the selected key which is invoking clear under selector right so when i run the code i have to get this particular output right server is listening so let us see whether it is waiting for the client or not

So, we will see the output. We will run the code in Java. So, I am running this code. Yes, server is listening. Yeah, this happens like what we had seen in Python or in C++, right.

So, now what we will do? We will see the path 2. Path 2 means the client 1, right. So, what is the output we are getting? So, here you see server is listening, right.

The server is listening output we are getting. So, to get the other output so, we are going to see right. So, this we had seen server is listening we are getting the output. So, let us wait we have to run the other part. So, then things will work right.

So, this I will start. So, now the server is alive right in the previous code server is alive. So, now we will check the client part right. So, as usual. io nio dot net everything is there right this is your importing here you have nio client class is nio client so this is the driver class for this program you have main

and here we are trying right so here the socket channel right which is invoking open so here you have the inet socket address the local host and the number right so this is creating the socket channel all right so this line number 10 which is creating a socket channel and connects it to the server listening on port 1 2 3 4 5 of the local host that is a meaning right i'm repeating again so this particular line number 10 which is creating a socket channel and connects it to the server listening on port 1 2 3 4 5 of the local host okay so line number 11 you have client which is invoking configure blocking

And then you are passing false so the meaning is it configures this particular command this particular syntax configures the channel to non-blocking mode all right so then you have the print statement so that means it is connecting to the server and then you have a message under string which is hello NIO server. So, this is not very difficult. Line number 17 you have object buffer under the class byte buffer.

So, this byte buffer that means the buffer is invoking wrap you are passing message you are passing message dot get bytes that means the message is invoking get bytes. So, that means it creates a byte buffer to hold the message bytes. right it is creating a byte buffer it is creating a byte buffer to hold the message bytes right so that is the meaning and then line number 18 we have the client which is invoking write method by passing buffer that means it is writing the message to the server it is writing the message to the server so now system.out.println message sent So, whatever be the message right.

So, that will be printed. So, here we have an object read buffer line number 21 under the class byte buffer right. So, here this readBuffer right. So, you have the allocate the same similar one we had seen previously. So, that means, it is allocating a byte buffer

With a capacity of 256 bytes. To store the incoming data. Right. So this is what the meaning. The maximum 256 bytes.

Right. To store the incoming data. So now while true. These four lines will be executed. That we will see.

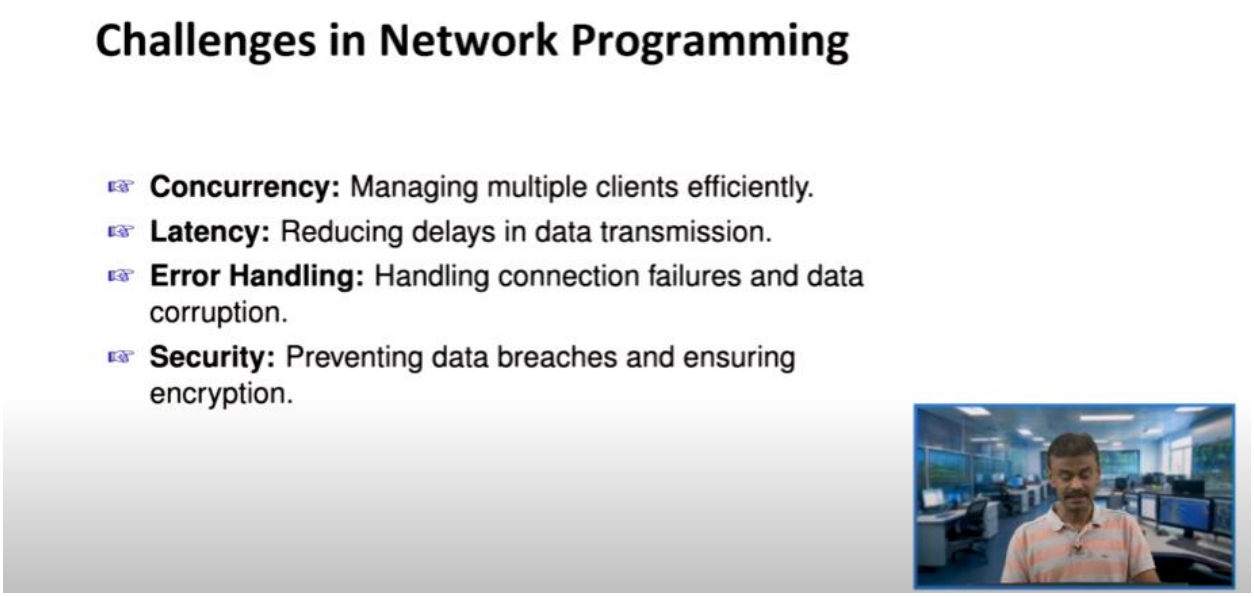
So you have bytes read. Which is a integer variable. Right. So the client which is invoking read function. By passing read buffer.

Right. So, it is reading. The data from the server channel into the read buffer, right? So, this is reading data from the server channel into the read buffer. If bytes read are greater than 0, which is positive, right?

So, it is checking if data was read successfully or not. The meaning here is, it is checking whether the data was read successfully or not. So, line number 25, we have the print statement, right? You have a print statement. So, receive a new string of readbuffer.ra 0 byte reads and trim.

That means it is printing the received data to the console, right? It is printing the received data to the console. That is the meaning, right? So, we have already seen that readbuffer.ra, right? And the trim, all right?

So, then you have The client will be closed if there is any exception, right? So, that will be caught over here, alright? Otherwise, if there is no exception and I run the code, I have to get the output, right? So, in the case of the client, So, let us see. That means it is connected to the server—the previous program, right? What we have not fixed will be run, right?

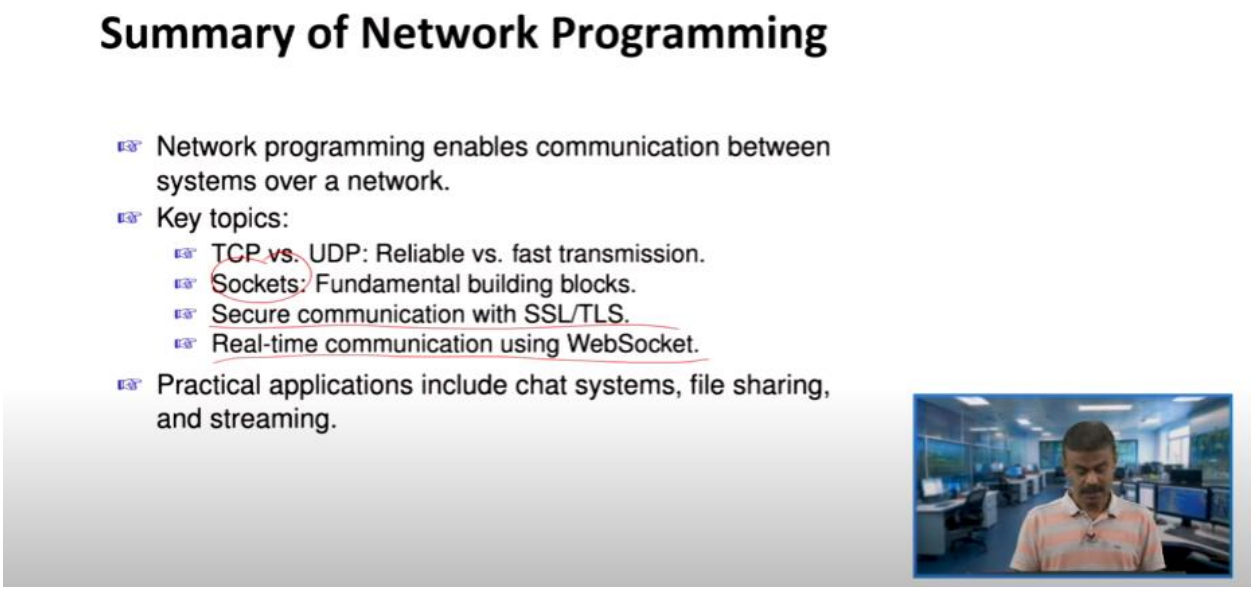


So, now practically, we will see. So, we will run the Java code. So, the client I am running, right? So, once it is established, So, we are seeing alright here—you go, right?

So, here we are seeing the output in the server. So, that means it is connected to the server. So, both I have open, and you can see the client—yes, connected to the server. Message sent to the NIO server, and here also we can see the output. Initially, we have only the server listening. Then, the client connected, and here you can see received hello-NIO service.

So, this is how we can establish the connection between the server and the client. So, this is with the help of Java. So, we are getting this output. So, we have seen in both client and server. In fact, this output also we had seen this is with the help of Java.

So, these are all some of the examples of the network applications right. So, you have a chat application. So, nowadays everyone is using right. one of the applications a chat application is example of a network applications file transfer applications either you have to download or upload all right or with the help of ftp you are transferring the file video streaming under udp right the first two are under tcp all right so this is under udp and iot applications right



So, when you want to communicate between two iot devices right so we say these are all under the network application and these are all some of the challenges we face in network programming concurrency so managing multiple clients efficiently that is a challenge reducing delays in data transmission many times it happens right so when you are having a video chat So, the data transmission, right, server side or the other side, you have some problem, then there will be a delay, right, in the data transmission. So, this is another challenge.

Error handling—more importantly, error handling, handling connection failures and data corruption, right? Which is a challenging task—and security: preventing data breaches and ensuring encryption, right? Ensuring the encryption. So, these are all the challenges we face in network programming. Right, and in network programming, we

know that it enables communication between systems over a network. So, you can have two, three, or more—all right? So, they will be connected, all right. So, for that, you require network programming, which we have studied, and then we talked about TCP versus UDP, right?

One is reliable versus fast transmission. We talked about sockets. Right, we talked about secure communication with SSL and TLS, right? So, these are all the key topics we learned—and real-time communication using WebSocket. We have seen several programs, right? So, from a practical applications point of view, what can you do when you are trying to—right? So, once you are going towards network programming— So, we can write—or you yourself can write—the program for chat systems, file sharing, streaming, etc.

Right. So, these are all—what do you mean by network programming? What are all the key topics that you have to study in network programming, and what are all the applications? Right. So, with this, I'm concluding this particular lecture.

Thank you very much.