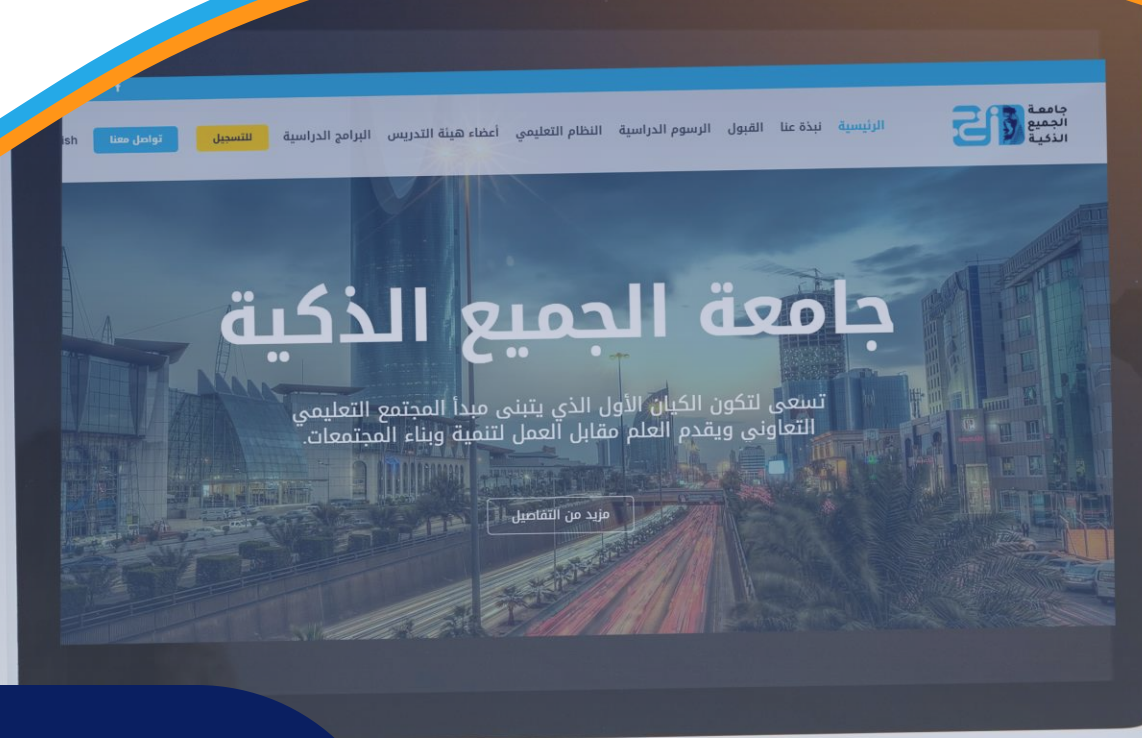




جامعة الجميع الذكية
EVERYONE'S SMART UNIVERSITY

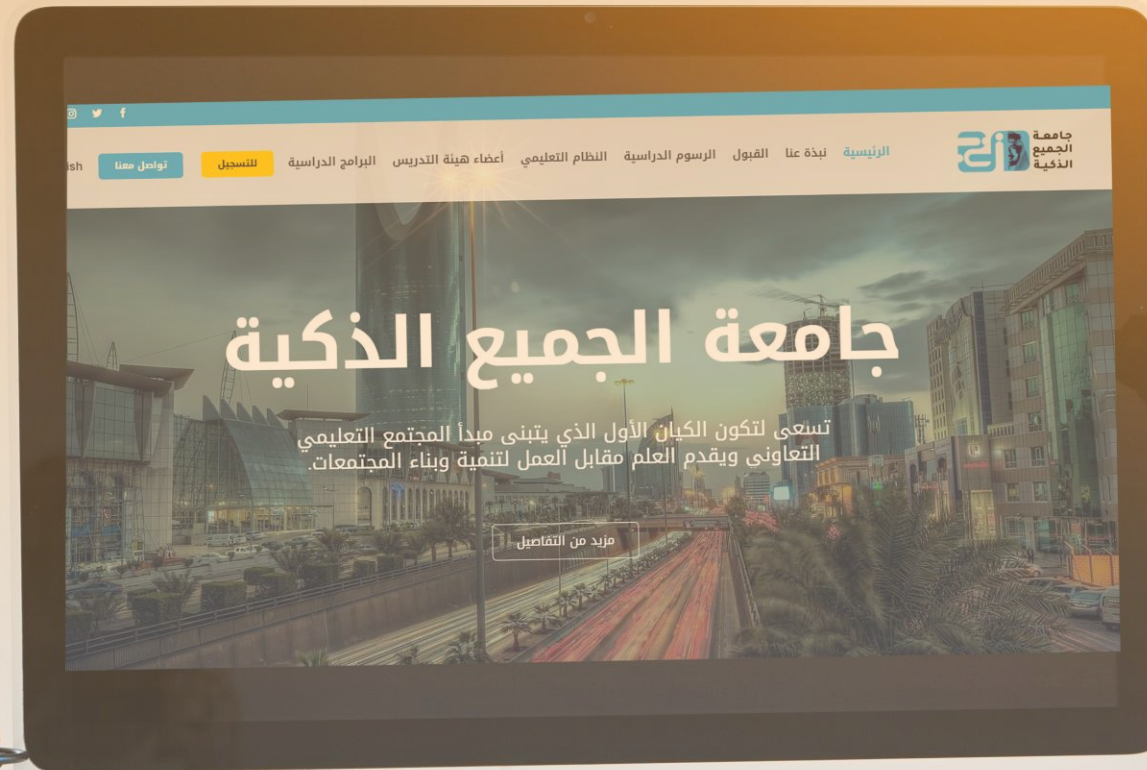


Prepared by
Dr. Hamada I. AbdulWakel
Lecture 4

CYB615
Network Programming



جامعة الجميع الذكية
EVERYONE'S SMART UNIVERSITY

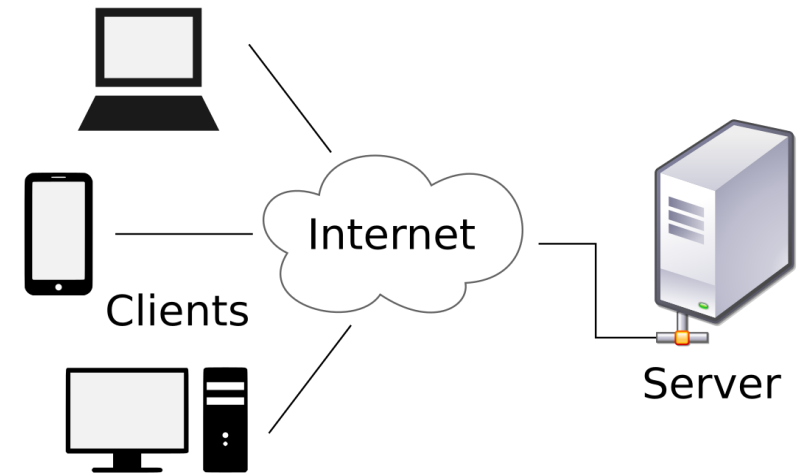


Module4: Socket Programming using Python – Part I

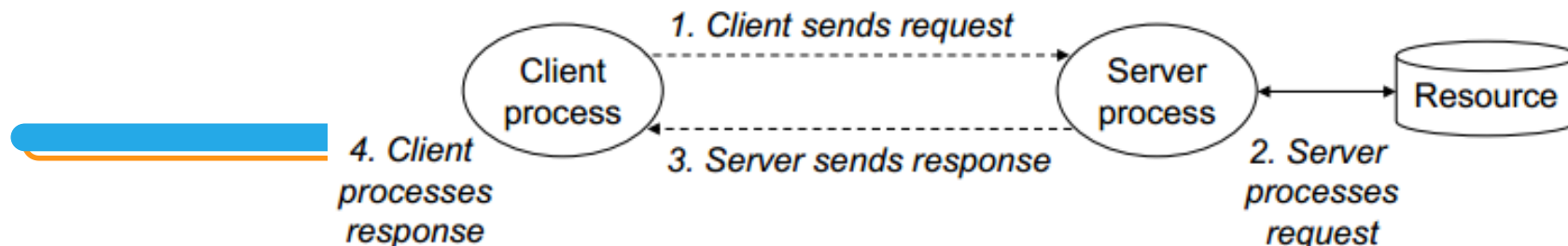
- The Client/Server Programming.
- Communication Protocols.
- TCP Socket Programming in Python.



- Every network application is based on the client-server model.
- An application consists of a server process and one or more client processes.
- A server manages some resource, and it provides some service for its clients by manipulating that resource.



- A client-server transaction consists of four steps.
 1. When a client needs service, it initiates a transaction by sending a request to the server.
 2. The server receives the request, interprets it, and manipulates its resources in the appropriate way.
 3. The server sends a response to the client, and then waits for the next request.
 4. The client receives the response and manipulates it.



- Although a computer usually has:
 - A single physical connection to the network.
 - Data could be sent by different applications to them do so through the use of ports configured on the same physical network connection.
- When data is to be transmitted over the internet to an application, it requires that we specify the address of the destination computer as well as the application's port number.
- A computer's address is a 32-bit IP address. The port number is a 16-bit number ranging from 0 to 65,535, with ports 0-1023 restricted by well-known applications like HTTP and FTP.



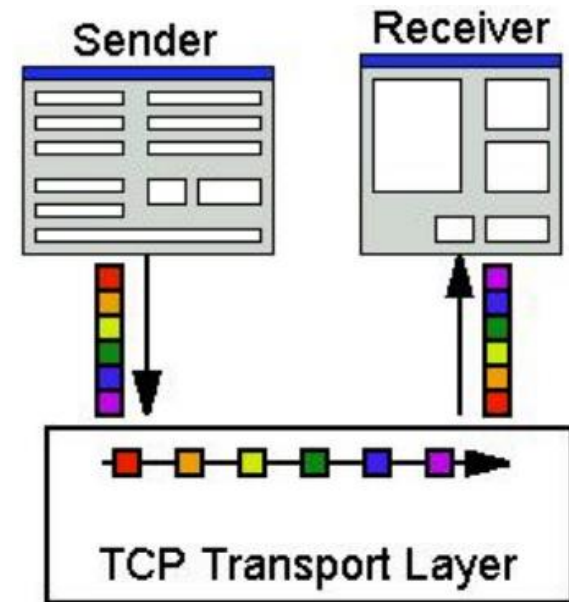
- Every packet (of data) has:
 - Source IP, Source Port, Destination IP, and Destination Port.
- Allows computing devices and applications to send data via a network and verify its delivery.
- There are two types of client-server application communication protocol:
 - Transmission Control Protocol (TCP).
 - User Datagram Protocol (UDP).



Lecture 4

Transmission Control Protocol (TCP)

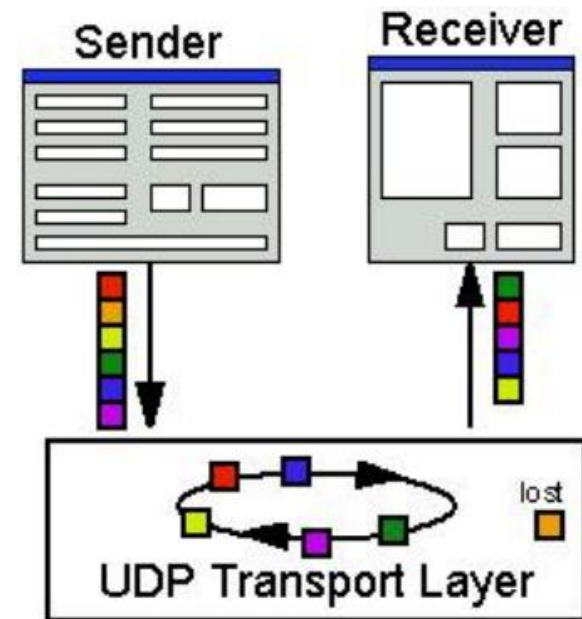
- A connection-oriented protocol that provides a reliable flow of data between end-to-end connection.
 - Three-way handshake. **"Wanna connect?" "Sure!" "Alright then!"**
- Slow overhead time of setting up.
- Retransmission.
 - Bigger overhead.
- In-order delivery.
- Congestion control.
 - Delays transmission when network is congested.
- Data doesn't always get sent out immediately (i.e., voice chat).



Lecture 4

User Datagram Protocol (UDP)

- Connection-less protocol that sends independent packets of data (datagrams), from one computer to another.
 - No connection to create and maintain.
- Faster since no overhead of setting up a connection.
- Small packet sizes.
- No compensation for lost packets.
- Packets can arrive out of order.
- No error recovery.
- No congestion control.



- Both protocols (UDP, TCP) are supported using "sockets".
 - A socket is a file-like object.
 - Allows data to be sent and received across the network like a file.
 - But it also includes functions to accept and establish connections.
- Before two machines can establish a connection, both must create a socket object.



- Socket basic methods:
 1. `socket()` to create a socket object.
 2. `gethostname()` to return a string containing the hostname.
 3. `gethostbyname()` to know the current machine's IP address.
 4. `connect()` to connect to a remote socket at an address.
 5. `bind()` to bind the socket to an address.
 6. `listen()` to listens for the connection made to the socket.
 7. `accept()` to accept a connection.
 8. `close()` to close the socket connection.



- To create a socket server, follow the following steps:
 1. Import socket module.
 2. Create a socket object.
 3. Get current machine name.
 4. Get port number for connection.
 5. Bind with the address.
 6. Determine number of allowed connections.
 7. Listen for connections.
 8. Connect and accept from clients.
 9. Send & receive messages.
 10. Close the connection.

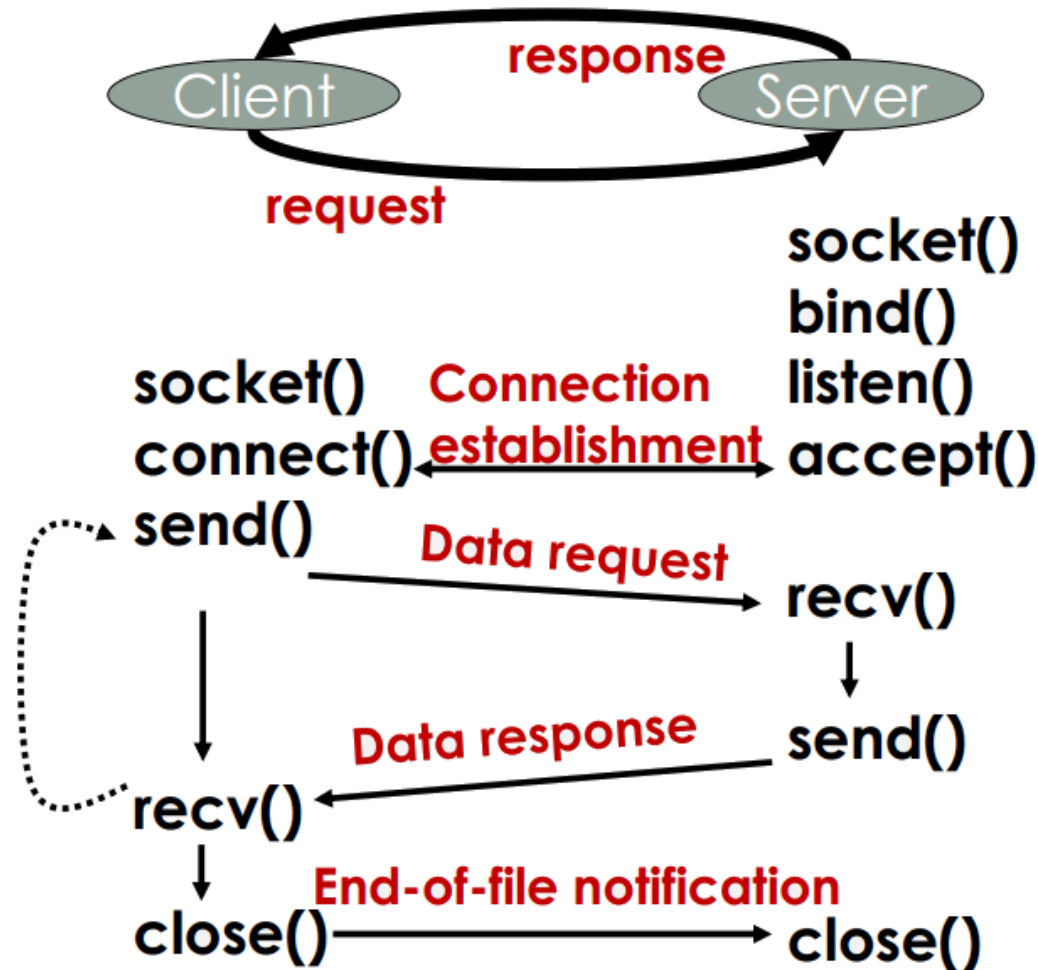


- To create a socket server, follow the following steps:
 1. Import socket module.
 2. Create a socket object.
 3. Identify the server machine name (i.e., hostname).
 4. Identify the server port number.
 5. Connect to the server.
 6. Send & receive messages.
 7. Close the connection.



Lecture 4

Major System Calls



Lecture 4

Working With TCP Socket Example

```
import socket
```

```
# create a TCP socket
```

```
soc = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

```
print("Socket successfully created!")
```



```
import socket
# create a TCP socket
try:
    soc = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    print("Socket successfully created!")
except socket.error as err:
    print("Failed to create a socket!")
    print(f"Reason: {err}")
```



Lecture 4

Working With TCP Socket Error Handling Example **Cont'd**

.....

```
target_host = input("Enter the target host name to connect: ")
target_port = input("Enter the target port: ")
try:
    sock.connect((target_host, int(target_port)))
    print(f"Socket Connected to {target_host} on port: {target_port}")
except socket.error as err:
    print(f"Failed to connect to {target_host} on port {target_port}")
    print("Reason: {str(err)}")
```



Lecture 4

Simple Server Socket Code

```
import socket
soc = socket.socket()
print("socket created")
# server and client in the same machine
soc.bind((socket.gethostname(), 5485))
soc.listen(3) #accepts three connection
print("Waiting for connections .....")
```



.....

while True:

#return client socket and address

clientSoc, clientAddre = soc.accept()

name = clientSoc.recv(1024).decode()

print("Connected with", clientAddre, name)

clientSoc.send(("Welcome we are connected :) " + name).encode())

clientSoc.close()



Lecture 4

Simple Client Socket Code

```
import socket
soc = socket.socket()
soc.connect((socket.gethostname(), 5485))
name = input("Enter your name? ")
soc.send(name.encode())
print(soc.recv(1024).decode())
soc.close()
```



- Exercise 1:

Using this link

<https://we-xpats.com/en/job/me/sa/search/?area=3411>

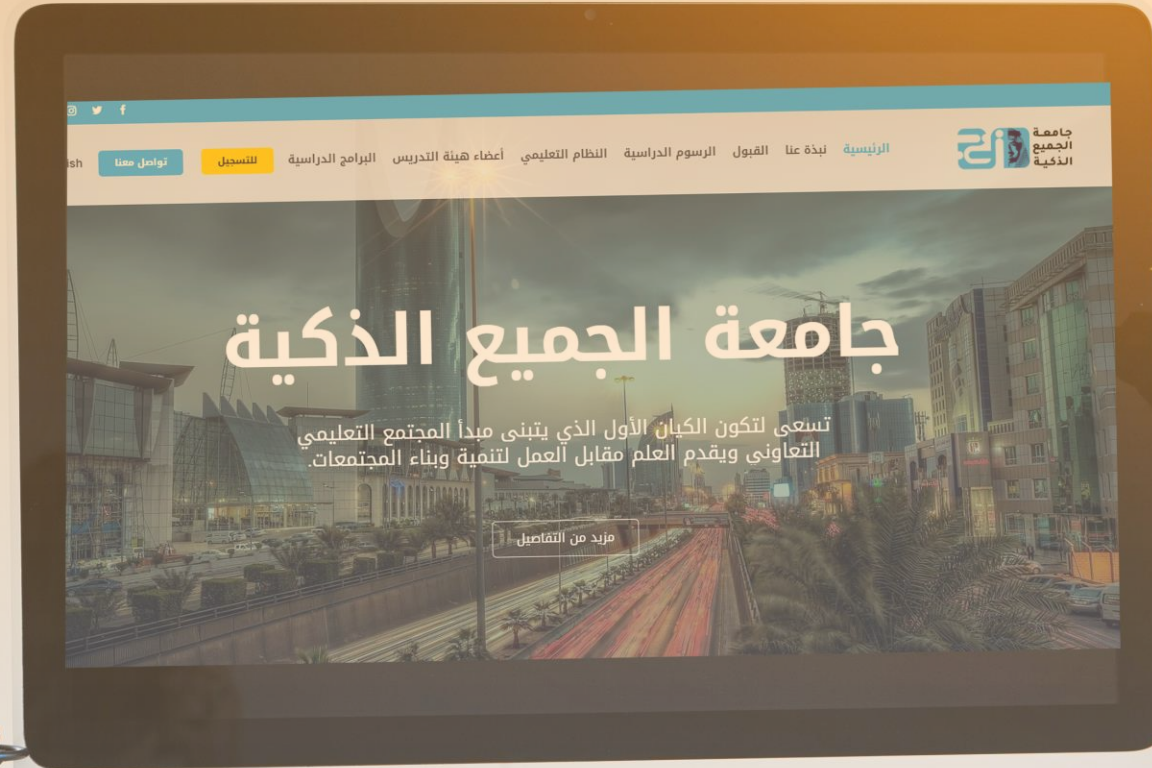
Create a web scraping application to get the available jobs in Makkah.



- Exercise 2:

Create a client/server application that pretends to be a calculator. It begins by asking the client about his name and age, then ask him to enter two numbers and the operation he want to perform. The server should calculate the result of the required operation and send it to the client.





جامعة الجميع الذكية
EVERYONE'S SMART UNIVERSITY

THANK YOU