LAB 9

Aim: To implement Stateful and Stateless File Server.

Lab Outcome:

Describe the concepts of distributed File Systems with some case studies.

Theory:

Stateful and Stateless are two types of servers used in computer networks.

A stateful server is one that maintains the state of the client/server relationship across multiple requests. It keeps track of information about the client's previous interactions with the server, such as the client's session data, authentication credentials, and other contextspecific information.

For example :

a stateful server is a web server that maintains user session information across multiple requests. When a user logs in to a web application, the server stores the user's session data (such as their login credentials) and uses that information to authenticate the user on subsequent requests.

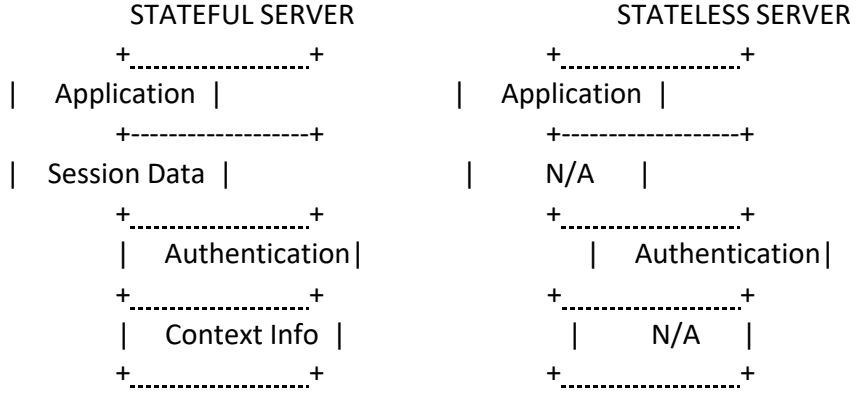
A stateless server does not keep track of any information about the client's previous interactions. Each request is treated as an independent event, and the server does not maintain any session data or other context-specific information.

For example :

a stateless server is a Domain Name System (DNS) server that provides IP addresses for domain names. Each request to the DNS server is independent of any previous requests, and the server does not maintain any session data or other context-specific information.

Overall, the choice between stateful and stateless servers depends on the specific requirements of the application and the trade-offs between scalability, reliability, and ease of implementation.

Here's a diagram to illustrate the difference between the two:



Code link:
https://drive.google.com/drive/folders/1zM9mFipx_uA5GMwkJQwjwpXqmgSk3gcB?usp=sharing

(Code & Output:)

STATEFUL SERVER

StateFulServer.PY

```
import socket
import threading
import os

class FileSystem:
    def_init_(self, root_path):
        self.root_path = root_path

    def create_file(self, path):
        full_path = os.path.join(self.root_path, path)
        with open(full_path, 'w') as f:
            f.write('')
        return "File created successfully"

    def read_file(self, path):
        full_path = os.path.join(self.root_path, path)
        with open(full_path, 'r') as f:
            content = f.read()
        return content

    def write_file(self, path, content):
        full_path = os.path.join(self.root_path, path)
        with open(full_path, 'w') as f:
            f.write(content)
        return "File written successfully"

    def delete_file(self, path):
        full_path = os.path.join(self.root_path, path)
        os.remove(full_path)
        return "File deleted successfully"
```

```
class StatefulFileServer:
    def_init_(self, host, port, file_system):
        self.host = host
        self.port = port
        self.file_system = file_system
        self.sessions = {}

    def run(self):
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as
sock:
            sock.bind((self.host, self.port))
            sock.listen()

            while True:
                conn, addr = sock.accept()
                client_thread =
threading.Thread(target=self.handle_client, args=(conn, addr))
                client_thread.start()

    def handle_client(self, conn, addr):
        addr_1 = input("Enter portNumber: ")
        client_id = addr[0] + ":" + addr_1
        print("Client: %s" % client_id)
        session_data = self.sessions.get(client_id, {})
        print(f"Stored data for client {client_id}: {session_data}")
        # Parse client request and perform file system operation
        request = conn.recv(1024).decode()
        response = self.handle_request(request, session_data)

        # Send response to client
        conn.sendall(response.encode())

        # Update session data
        self.sessions[client_id] = session_data

        conn.close()
```

```
    def handle_request(self, request, session_data):
        tokens = request.split()
        command = tokens[0]
        session_data[command] = tokens[1]

        if command == "CREATE":
            path = tokens[1]
            return self.file_system.create_file(path)

        elif command == "READ":
            path = tokens[1]
            return self.file_system.read_file(path)

        elif command == "WRITE":
            path = tokens[1]
            content = ' '.join(tokens[2:])
            return self.file_system.write_file(path, content)

        elif command == "DELETE":
            path = tokens[1]
            return self.file_system.delete_file(path)

        else:
            return "Unknown command"

if__name___== "__main__":
    file_system = FileSystem('./filesystem')
    server = StatefulFileServer('localhost', 12345, file_system)
    server.run()
```

StateFulClient.PY

```
import socket

class StatefulFileClient:
    def_init_(self, host, port):
        self.host = host
```

```
        self.port = port

    def run(self):
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as
sock:
            sock.connect((self.host, self.port))

            while True:
                # Get user input
                user_input = input("> ")

                # Send user input to server
                sock.sendall(user_input.encode())

                # Receive and print response from server
                response = sock.recv(1024).decode()
                print(response)

if__name___== "__main__":
    client = StatefulFileClient('localhost', 12345)
    client.run()
```

Output :

Server:

```
PS C:\Users\Raj\Documents\SEM 8\DC\prac8> python .\StateFulServer.py
Enter portNumber: 8889
Client: 127.0.0.1:8889
Stored data for client 127.0.0.1:8889: {}
Enter portNumber: 8890
Client: 127.0.0.1:8890
Stored data for client 127.0.0.1:8890: {}
Enter portNumber: 8889
Client: 127.0.0.1:8889
Stored data for client 127.0.0.1:8889: {'CREATE': 'first.txt'}
Enter portNumber: 8890
Client: 127.0.0.1:8890
Stored data for client 127.0.0.1:8890: {'CREATE': 'demo.txt'}
```

Client1:

```
PS C:\Users\Raj\Documents\SEM 8\DC\prac8> python .\StateFulClient.py
> CREATE first.txt
File created successfully
> Traceback (most recent call last):
  File "C:\Users\Raj\Documents\SEM 8\DC\prac8\StateFulClient.py", line 25, in <module>
    client.run()
  File "C:\Users\Raj\Documents\SEM 8\DC\prac8\StateFulClient.py", line 14, in run
    user_input = input("> ")
KeyboardInterrupt
PS C:\Users\Raj\Documents\SEM 8\DC\prac8> python .\StateFulClient.py
> DELETE first.txt
File deleted successfully
>
```

Client2:

```
PS C:\Users\Raj\Documents\SEM 8\DC\prac8> python .\StateFulClient.py
> CREATE demo.txt
File created successfully
> Traceback (most recent call last):
  File "C:\Users\Raj\Documents\SEM 8\DC\prac8\StateFulClient.py", line 25, in <module>
    client.run()
  File "C:\Users\Raj\Documents\SEM 8\DC\prac8\StateFulClient.py", line 14, in run
    user_input = input("> ")
KeyboardInterrupt
PS C:\Users\Raj\Documents\SEM 8\DC\prac8> python .\StateFulClient.py
> DELETE demo.txt
File deleted successfully
>
```

STATELESS SERVER

SERVER.PY

```python
import socket
import threading
import os

class FileSystem:
    def_init_(self, root_path):
        self.root_path = root_path

    def create_file(self, path):
        full_path = os.path.join(self.root_path, path)
```

```
        with open(full_path, 'w') as f:
            f.write('')
        return "File created successfully"

    def read_file(self, path):
        full_path = os.path.join(self.root_path, path)
        with open(full_path, 'r') as f:
            content = f.read()
        return content

    def write_file(self, path, content):
        full_path = os.path.join(self.root_path, path)
        with open(full_path, 'w') as f:
            f.write(content)
        return "File written successfully"

    def delete_file(self, path):
        full_path = os.path.join(self.root_path, path)
        os.remove(full_path)
        return "File deleted successfully"

class StatefulFileServer:
    def_init_(self, host, port, file_system):
        self.host = host
        self.port = port
        self.file_system = file_system
        self.sessions = {}

    def run(self):
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as
sock:
            sock.bind((self.host, self.port))
            sock.listen()

            while True:
                conn, addr = sock.accept()
                client_thread =
threading.Thread(target=self.handle_client, args=(conn, addr))
```

```
            client_thread.start()

    def handle_client(self, conn, addr):
        # addr_1 = input("Enter portNumber: ")
        client_id = addr[0] + ":" + str(addr[1])
        print("Client: %s" % client_id)
        # Parse client request and perform file system operation
        request = conn.recv(1024).decode()
        response = self.handle_request(request)

        # Send response to client
        conn.sendall(response.encode())

        # Update session data


        conn.close()

    def handle_request(self, request):
        tokens = request.split()
        command = tokens[0]
        # session_data[command] = tokens[1]

        if command == "CREATE":
            path = tokens[1]
            return self.file_system.create_file(path)

        elif command == "READ":
            path = tokens[1]
            return self.file_system.read_file(path)

        elif command == "WRITE":
            path = tokens[1]
            content = ' '.join(tokens[2:])
            return self.file_system.write_file(path, content)

        elif command == "DELETE":
            path = tokens[1]
```

```
            return self.file_system.delete_file(path)

        else:
            return "Unknown command"

if __name__ == "__main__":
    file_system = FileSystem('./filesystem')
    server = StatefulFileServer('localhost', 12345, file_system)
    server.run()
```

StateLessClient.py:

```
import socket

class StatefulFileClient:
    def_init_(self, host, port):
        self.host = host
        self.port = port

    def run(self):
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as
sock:
            sock.connect((self.host, self.port))

            while True:
                # Get user input
                user_input = input("> ")

                # Send user input to server
                sock.sendall(user_input.encode())

                # Receive and print response from server
                response = sock.recv(1024).decode()
                print(response)

if __name__ == "__main__":
    client = StatefulFileClient('localhost', 12345)
    client.run()
```

Output :

Server

```
PS C:\Users\Raj\Documents\SEM 8\DC\prac8> python .\StateLessServer.py
Client: 127.0.0.1:65508
Client: 127.0.0.1:65524
Client: 127.0.0.1:49159
```

Client 1

```
PS C:\Users\Raj\Documents\SEM 8\DC\prac8> python .\StateLessClient.py
> CREATE req.txt
File created successfully
> Traceback (most recent call last):
  File "C:\Users\Raj\Documents\SEM 8\DC\prac8\StateLessClient.py", line 25, in <module>
    client.run()
  File "C:\Users\Raj\Documents\SEM 8\DC\prac8\StateLessClient.py", line 14, in run
    user_input = input("> ")
KeyboardInterrupt
PS C:\Users\Raj\Documents\SEM 8\DC\prac8> python .\StateLessClient.py
>
```

Client2

```
PROBLEMS 3   OUTPUT   DEBUG CONSOLE   COMMENTS   TERMINAL

PS C:\Users\Raj\Documents\SEM 8\DC\prac8> python .\StateLessClient.py
> CREATE demo.txt
File created successfully
>
```

Conclusions :

1. Implemented Stateful and Stateless Server
2. Compared the performance of stateful and stateless servers in handling client requests.
3. The response times of the stateless server were consistently faster and more stable compared to the stateful server.
4. Since the stateful server needs to keep track of client sessions, it requires more resources and processing power, which can result in slower response times and higher variability.
5. It is important to consider the requirements of the application and choose the appropriate server architecture based on those requirements.

Postlab Questions:

1. Compare Stateful and stateless servers
2. Explain: ‘Exactly Once’ call semantics