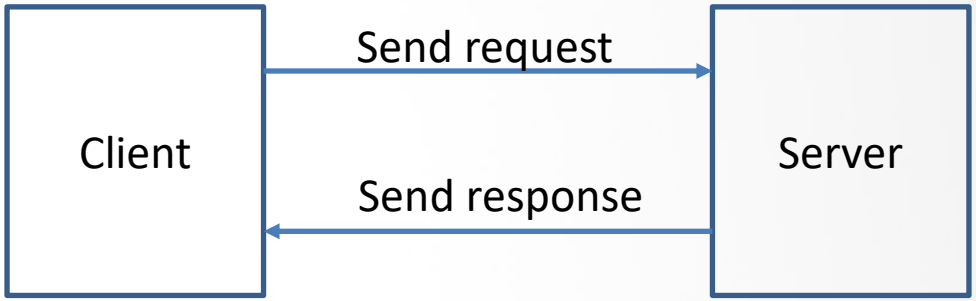
Tutorial 5
TCP server/client

Kai CHEN

C-S Model

Client-Server Model



- Send requests
- Receive, further actions

- Wait for requests (loop)
- Receive, process, response

Socket

Socket object

- Python’s built-in module
- Low-level networking interface
- Communication between processes (on different hosts)
- Identified by **IP address** and **port**
- Implement various socket system calls

Socket

Socket methods - create

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

IPv4

TCP

Socket

Socket methods - bind

```
server_socket.bind(("localhost", 50000))

server_socket.bind(("127.0.0.1", 50000))

server_socket.bind((socket.gethostname(), 50000))
```

tuple: (address, port)

Socket

Socket methods - listen

```
server_socket.listen(1)
```

max waiting clients

Socket

Socket methods - connect

```
client_socket.connect(("localhost", 50000))
```

address port

Socket methods - accept

```
(client_socket, address) = server_socket.accept()
```

wait until a client tries to connect

Socket

Socket methods - recv

```
data = client_socket.recv(1024)
```

received bytes

buffer size

Socket

Socket methods - send

```
length = client_socket.send(data)
```

sent bytes

Socket methods - sendall

```
client_socket.sendall(data)
```

Socket methods - close

```
client_socket.close()
```

Example

Server

1. Create
2. Bind
3. Listen
4. Loop:
    1. Accept
    2. Receive (optional)
    3. #processing#
    4. Send
    5. Close

Client

1. Create
2. Connect
3. Send
4. #Wait for the server#
5. Receive (optional)
6. Close

Example

Simple client and server

```python
import socket

# create an INET socket
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# bind the socket to the host and a port
server_socket.bind(("localhost", 50000))

# Listen for incoming connections from clients
server_socket.listen(1)

# A indefinite loop
while True:
    # accept connections from outside
    (client_socket, address) = server_socket.accept()
    print("accepted connection from %s" % str(address))

    # Read data from client and send it back
    data = client_socket.recv(1024)
    print("Received %s from %s" % (data.decode("utf-8"), address))
    client_socket.sendall(data)

    # Close the socket
    client_socket.close()
```

Example

Simple client and server

```python
import socket

# create an INET TCP socket
soc = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# connect to the server (change localhost to an IP address if necessary)
soc.settimeout(5)
soc.connect(("localhost", 50000))
print("Connected to server")

# Send a message to the server
msg = "Hello Server!".encode("utf-8")
soc.send(msg)

# Receive data from the server
data = soc.recv(1024)
print("Sent {} to server.".format(data.decode("utf-8")))

# Always close the socket after use
soc.close()
```

Example

Outputs

server

```
$ python simple_server.py
accepted connection from ('127.0.0.1', 56486)
Received Hello Server! from ('127.0.0.1', 56486)
```

client

```
$ python simple_client.py
Connected to server
Sent Hello Server! to server.
```

Example

Send image data to server

encode image bytes

append a special string

```
img = Image.open("test.png")
buffered = BytesIO()
img.save(buffered, format="PNG")
encoded_img = base64.b64encode(buffered.getvalue())
```

```
data = {
    'image': encoded_image.decode("utf-8"),
    'other_info': xxxx
}
data = json.dumps(data) + "##END##"
soc.sendall(data.encode("utf-8"))
```

Example

Receive encoded image data

receive until the special string

decode the image data

save to image

```
data = ""
while True:
    part = client_socket.recv(1024)
    data = data + part.decode("utf-8")
    if "##END##" in data:
        # Delimiter is received, we can stop receiving
        break

data = data.replace("##END##", "")
data = json.loads(data)
image_data = base64.b64decode(data['image'])
with open('image.png', 'wb') as f:
    f.write(image_data)
```

Thank you!