Practical Work 1

1. Protocol Specification

1. **Connection Initiation**
2. **Filename Transmission**
3. **Acknowledgement (ACK)**
4. **Data Transmission**
5. **Connection Termination**

```
participant Client
participant Server

Client->>Server: Initiate Connection
Client->>Server: Transmit Filename
Server->>Client: Transmit "ACK" Signal
Client->>Server: Transmit File Data (Segments)
Client->>Server: Terminate Connection
```

2. System Architecture

The system architecture is composed of two distinct, interacting entities. The server functions as a passive entity, binding to a specific port and awaiting incoming requests. Conversely, the client functions as the active entity, responsible for initiating the connection and driving the transfer process.

```
S1[Initialize Server] --> S2[Bind Port & Listen]
S2 --> S3[Accept Connection]
S3 --> S4[Receive Data]
end

C1[Initialize Client] --> C2[Connect to Server Address]
C2 --> C3[Transmit Data]
end

C2 -- Network Connection --> S3
C3 -- File Transfer --> S4
```

3. Implementation Details

The implementation utilizes the Python standard socket library, selected for its comprehensive networking capabilities and ubiquity, thereby obviating the necessity for external dependencies. The codebase is segregated into two distinct modules: the client-side transmission script and the server-side reception script.

(client.py)

This module is responsible for reading the binary data from the local file system and transmitting it over the network. Pre-emptive exception handling mechanisms are employed to validate the source file's existence prior to execution.

```
import socket
import sys
import os

def send_file(server_ip, port, filename):
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    try:
        server_address = (server_ip, port)
        print(f'Initiating connection to {server_ip} on port {port}...')
        client_socket.connect(server_address)

        print(f'Transmitting filename: {filename}')
        client_socket.sendall(filename.encode('utf-8'))

        # Await Server Acknowledgement (ACK)
        ack = client_socket.recv(1024)
        if ack.decode('utf-8') != 'ACK':
            print('Error: Server acknowledgement was not received.')
            return

        print('Server acknowledgement received. Commencing data transmission...')

        with open(filename, 'rb') as f:
            while True:
                data = f.read(1024)
                if not data:
                    break
                client_socket.sendall(data)

        print('File transmission completed successfully.')
    except Exception as e:
        print(f"Critical Error: {e}")
    finally:
        client_socket.close()

if __name__ == '__main__':
    if len(sys.argv) < 4:
        print(f"Usage: {sys.argv[0]} <server_ip> <port> <filename>")
        sys.exit(1)

    server_ip = sys.argv[1]
    port = int(sys.argv[2])
    filename = sys.argv[3]

    if not os.path.exists(filename):
        print(f"Error: The file '{filename}' could not be located.")
        sys.exit(1)

    send_file(server_ip, port, filename)
```

(server.py)

This module maintains a listening state, responding to connection requests with an acknowledgement signal, and subsequently writing the received data stream to the local storage medium.

```
import socket
import sys
import os

def start_server(port):
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    server_address = ('', port)
    print(f'Server initialized and listening on port {port}...')
    server_socket.bind(server_address)

    server_socket.listen(1)

    while True:
        print('Awaiting incoming connections...')
        connection, client_address = server_socket.accept()
        try:
            print(f'Connection established with {client_address}')

            data = connection.recv(1024)
            if data:
                filename = data.decode('utf-8')
                print(f'Incoming file transfer: {filename}')

                # Transmit ACK to signal readiness
                connection.sendall(b'ACK')

                with open(filename, 'wb') as f:
                    while True:
                        data = connection.recv(1024)
                        if not data:
                            break
                        f.write(data)

                print(f'File {filename} received successfully.')

        except Exception as e:
            print(f"System Error: {e}")
        finally:
            connection.close()

if __name__ == '__main__':
    if len(sys.argv) < 2:
        print(f"Usage: {sys.argv[0]} <port>")
        sys.exit(1)

    port = int(sys.argv[1])
    start_server(port)
```