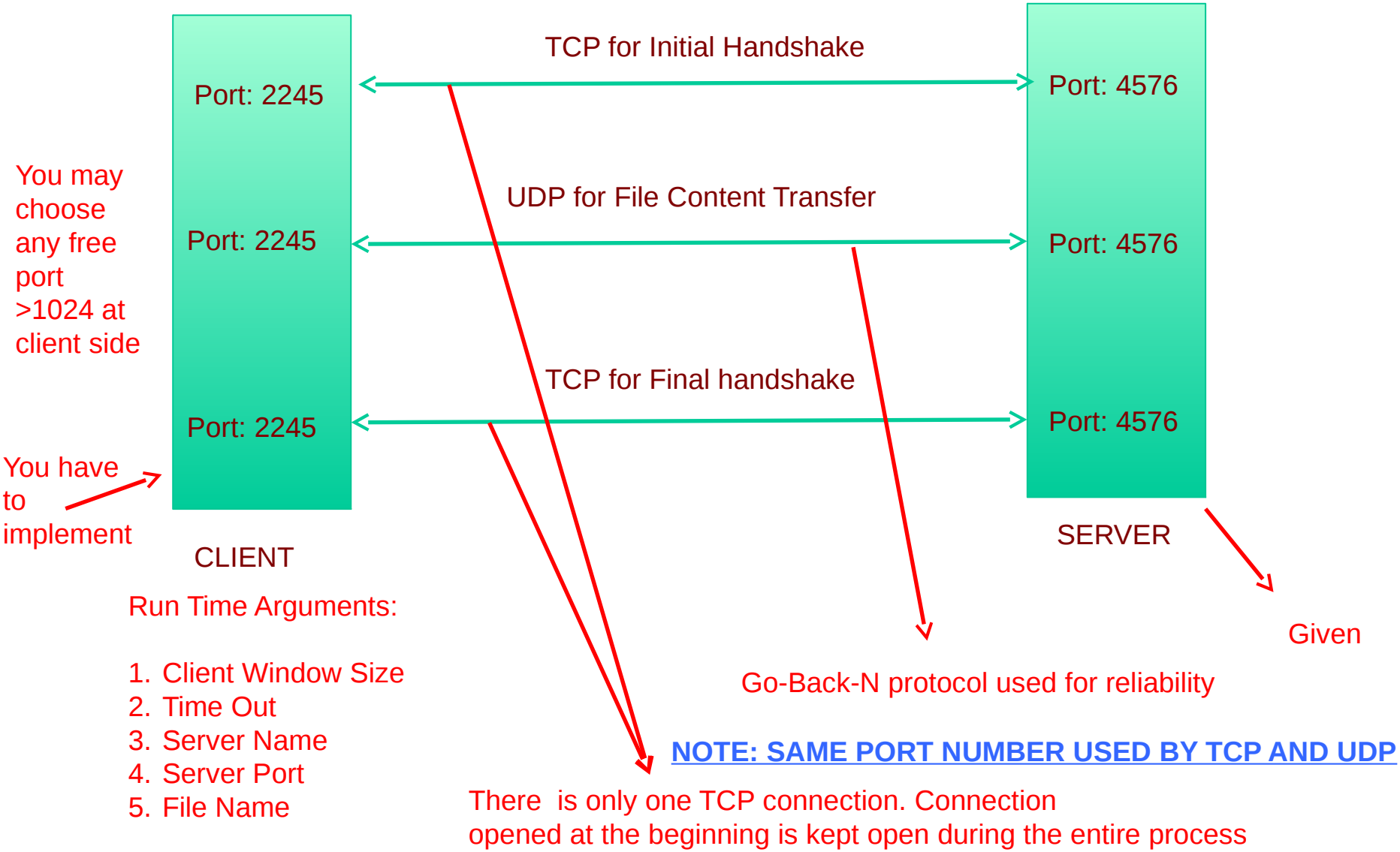
CPSC 441
Assignment-3 Discussion

Department of Computer Science
University of Calgary

Reference: Assignment description and design notes posted in D2L by Dr. Majid

Overview of FastFTP protocol



How to get TCP Local Port Number for Creating UDP Socket

Create TCP Socket

```
Socket socket = new Socket(''localhost'', 8888);
```

Create UDP Socket with same local port number as used by TCP

```
DatagramSocket clientSocket = new DatagramSocket(socket.getLocalPort());
```

Algorithm at Client (A high level description of what you need to implement)

- Step-1: Open a TCP connection
- Step-2: Send file name to the server over TCP
- Step-3: Wait for server response over TCP
- Step-4: Open a UDP socket
- Step-5: Send file content to server as UDP segments
- Step-6: Send an end of Transmission message to server over TCP
- Step-7 Clean up and close TCP/UDP sockets

Note: All communication (TCP and UDP) in binary format

Writing/Reading from TCP Socket

- Use Java stream classes
  - DataInputStream
  - DataOutputStream

Writing to TCP Socket: File Name

```
DataOuputStream outputstream = new DataOuputStream (socket.getOutputStream());


String Filename = ”testfile";

try
{
      outputStream.writeUTF(Filename);
outputStream.flush();

}
catch (IOException e)
{

// code for Exception Handling

}
```

Note the stream type used

Note that file name is a run time argument

First 2 bytes is the string length. The string itself follows later in UTF-8 encoding format (**Step 2**)

Make sure that “filename” is indeed send

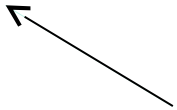
Writing to TCP Socket: End of Transmission

```
try
{
        outputStream.writeByte(0);
outputStream.flush();

}
catch (IOException e)
{

// code for Exception Handling

}
```

End of transmission message (**Step-6**)

Reading from TCP Socket: Server Response

```
DataInputStream inputstream = new DataInputStream(socket.getInputStream());

try
{
        byte respcode = inputstrean.readByte();

}

catch (IOException e)
{

//Exception handling

}
```

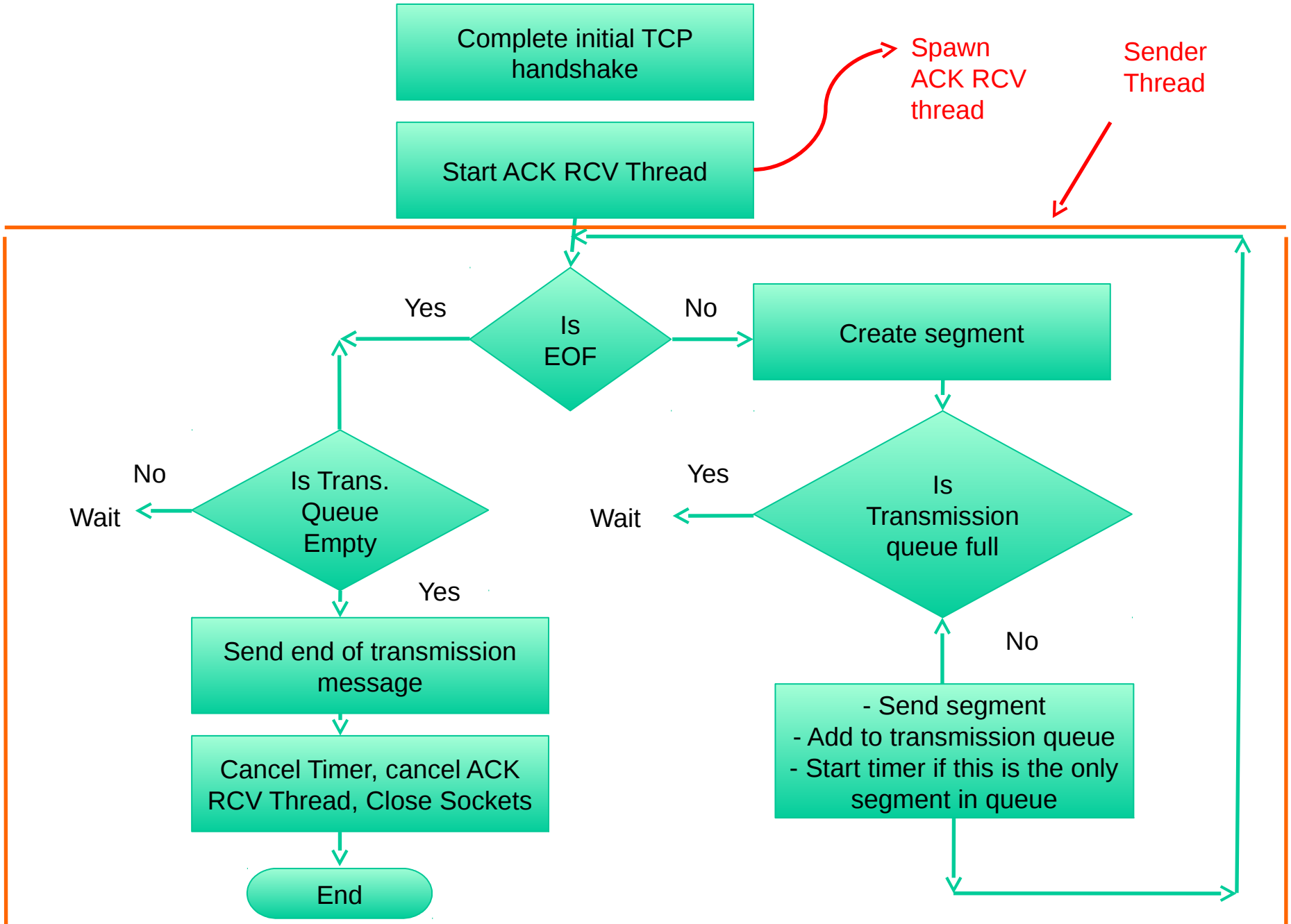


Server response during initial handshake (**Step-3**)

Client Program Structure

- Following slides will give you (hopefully!) an idea on how to implement step-5 of the algorithm.

Disclaimer: This is based on my own interpretation of design note posted in D2L. Following implementation may not be the simplest or best


Complete initial TCP handshake
Start ACK RCV Thread
Spawn ACK RCV thread
Sender Thread
Is EOF
Yes
No
Create segment
Is Trans. Queue Empty
No
Wait
Yes
Send end of transmission message
Cancel Timer, cancel ACK RCV Thread, Close Sockets
End
Is Transmission queue full
Yes
Wait
No
- Send segment
- Add to transmission queue
- Start timer if this is the only segment in queue

Send Thread (Main thread): sending file content

- Send segment
- Add segment to transmission queue
- If this segment is the only segment in queue, start the timer (see slide on "timer")

```
 /* logic for sending segment */
try
{
1. create segment from payload
2. create datagram from segment
3. send datagram
}
Catch (Exception e)
{

}
```

ACK Receive Thread

- Cancel Timer if ACK is valid
- Get ACK number from UDP packet
- While(txQueue.element().getSeqNum() < ackseg.getSeqNum())
  - txQueue.remove()
- If queue not empty, start timer

Note: Same UDP socket should be used for sending messages to the server and receiving ACKs from the server

Byte array to receive a Datagram packet.

Receiving ACK

```
DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);

clientSocket.receive(receivePacket);

Segment ackseg = new Segment(receiveData);

acknum = ackseg.seqNum();
```

Convert the datagram packet received into segment format

Get ack number from the segment

Timer Thread

- Use Timer class
- Need not create thread explicitly – Java does this in the background
- Use Timer class

Create timer thread

```
Timer timer = new Timer(true)

timer.schedule(new TimeoutHandler(),
1000)
```

Schedule timer to go off in 1000 ms.
Execute the **run** method in 'TimeoutHandler'

Similar to thread creation!

```
class TimeoutHandler extends TimerTask
{

   // define constructor

public void run()
{
      // call method to process time out as
described in the next slide
}
```

Steps to Process Time-outs

1. Get list of all pending segments from transmission queue

```
Segment[ ] pending_seg;
pending_seg = txQueue.toArray() /* List of packets still in transmission
                                   queue */
```

2. Send all the packets still in transmission queue

3. If queue is not empty, start the timer