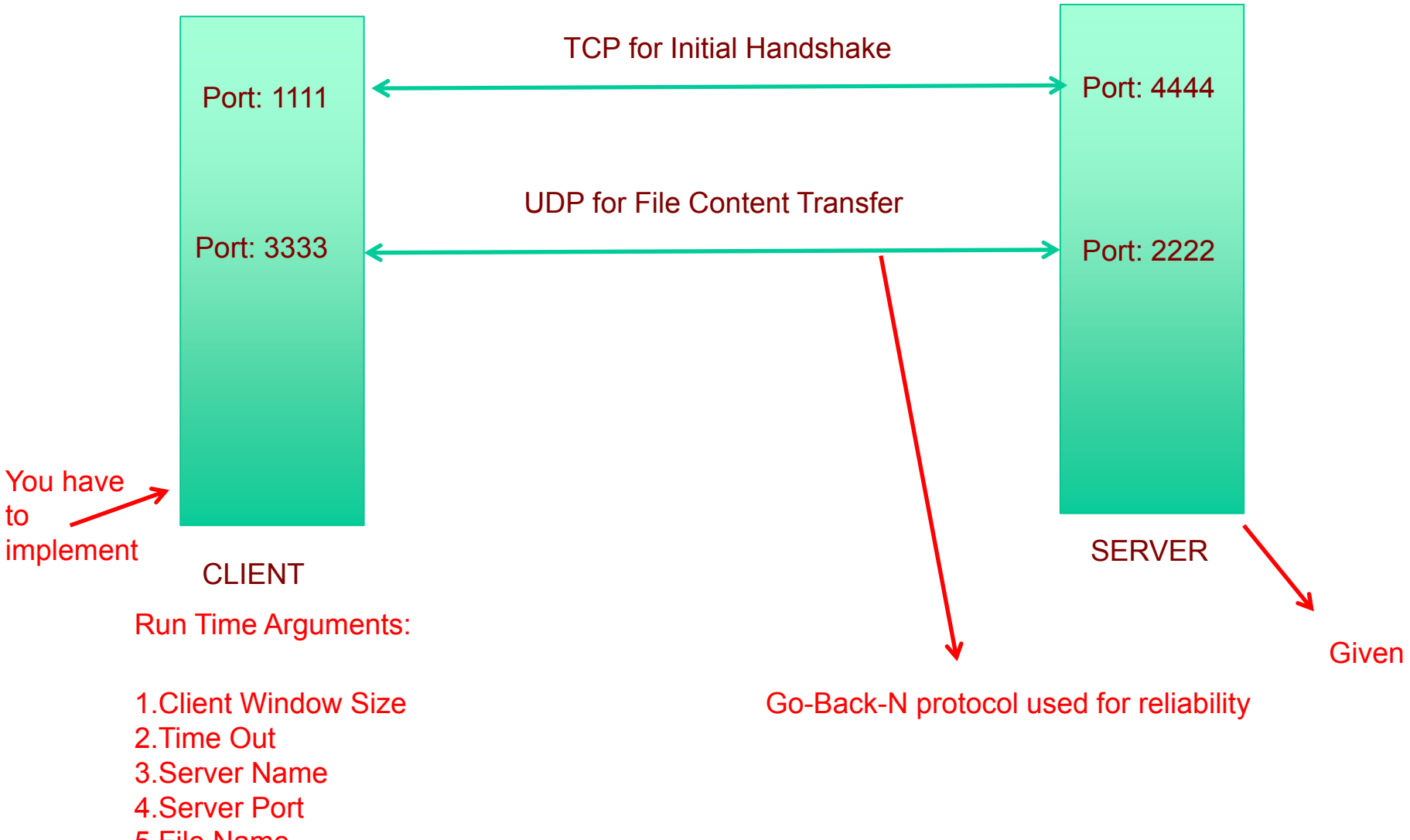
CPSC 441
Assignment-3 Discussion

Department of Computer Science
University of Calgary

Overview of FastFTP protocol



Run Time Arguments:

1. Client Window Size
2. Time Out
3. Server Name
4. Server Port
5. File Name

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

- Step-1: Open a TCP connection
- Step-2: Open a UDP socket
- Step-2: Send file name, file length, local UDP port number to the server over TCP
- Step-3: Read server UDP port number over TCP
- Step-5: Send file content to server as UDP segments
- 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: Local UDP Port Number

```
try
{
        outputStream.writeInt(udpSocket.getLocalPort());
        outputStream.flush();

}
catch (IOException e)
{

        // code for Exception Handling

}
```

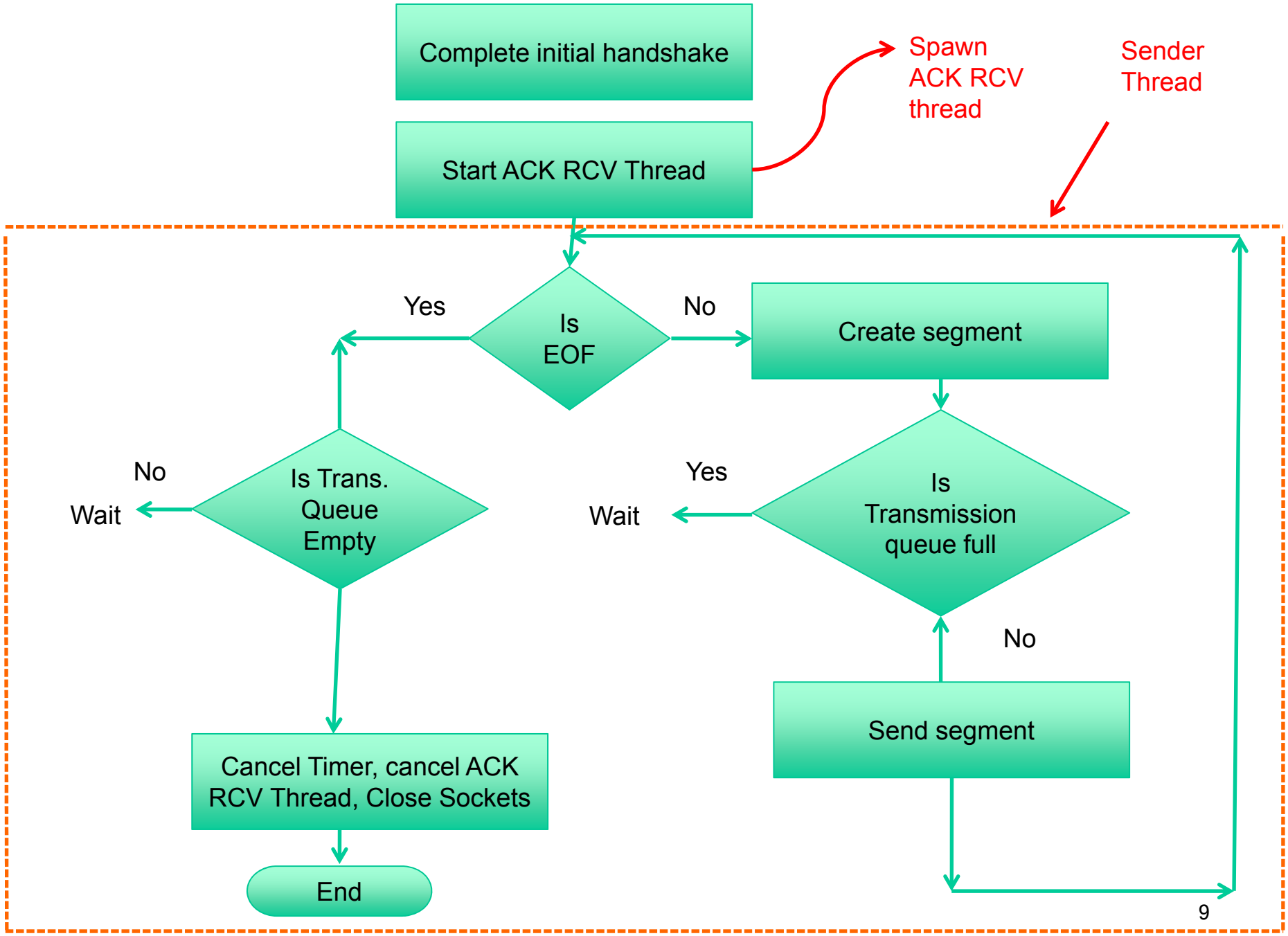
Reading from TCP Socket: Server UDP Port Number

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

try
{
        int remoteUdpPort = inputstrean.readInt();

}

catch (IOException e)
{

//Exception handling

}
```

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 handshake
Start ACK RCV Thread
Spawn ACK RCV thread
Sender Thread
Is EOF
Yes
No
Create segment
Is Trans. Queue Empty
No
Wait
Is Transmission queue full
Yes
Wait
No
Send segment
Cancel Timer, cancel ACK RCV Thread, Close Sockets
End

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”)

Sending data segment

```
try
{

Segment segment_send = new Segment(seqnum, payload)
DatagramPacket sendPacket =
        new DatagramPacket(segment_send.getBytes(),
segment_send.getBytes().length, ServerIP, ServerPort);

clientSocket.send(sendPacket);

}
catch(Exception e)
{
      // handle exception
}
```

Byte array containing part of file content

Type InetAddress: Refer UDP client code

Number of bytes to write

Convert segment to byte array

ACK Receive Thread

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

- ❖Cancel Timer if ACK is valid
- ❖Get ACK number from UDP packet
- ❖While (SeqNum of txQueue.element() < acknum)
  remove the segment at the head of queue
- ❖If queue not empty, start timer

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
}
```

Steps to Process Time-outs

1. Get list of all pending segments from transmission queue

```
Segment[ ] pending_seg;
pending_seg = txQueue.toArray()
```

2. Send all segments in pending_seg

3. if queue is not empty, start the timer