Assignment 4

CPSC 441

Oscar Campos 30057153

How I implemented re-transmission was a part of the Timer Task. Re-transmission only occurs when the timer timeouts and we retransmit the current sequence number with its payload. How I set up the timer was like a thread class. I first implemented the time class that extends to Timer Task. With in this class there is a constructor that takes the UDP socket, segment sent, packet, and the sequence number that we use to retransmit the package. When a time out happens the run method of the class will run printing to the screen that a time out has occur and that a retransmission of the packet was sent once again to the server. A copy of the segment that is being transmitted is also within the class this is because like mention in the notes and the assignment that we want to avoid any race condition. A race condition is when two threads access a shared variable at the same time but since we have a copy of the segment in the timer class we can avoid this without having a synchronized block that deals with this type of problem. Now with that I must explain how my Timer class works. Let's now look at when I call it, when I start it, and when I close it. The time class is called in the second part of this assignment after the handshake has been established. We first call it when we send the encapsulated segment to the server. A TimerTask is created with the following information, UDP socket, Segment, packet, and sequence number. I then called scheduleAtFixedRate to avoid having to reset the timer, like mention in the notes provided by the professor. That being said we pass it the NewTask which is the TimerTask and the Timeout provided by the code. Now that the timer is running whenever time out happens a retransmission will be sent to the server with the same sequence number and package. If a transmission does not happen it is because the next sequence number recived by the UDP socket from the server was the correct one. This means that we send sequence number n, and we received sequence number n+1 and in this case we don't need to retransmit the previous segment because this how we want the stop and wait protocol to work, and we call NewTask.cancel() in order to cancel the timer class for this sequence number. Once the file has been fully transmitted to the server we then close all the timer objects using .close() and .purge() this happens after my while loop that reads the file. What happens if the implementation if a timeout occurs at the same time that an ACK arrives, race condition? This can happen because the segment might get retransmitted before it was initialized by the main program. Like mention earlier in the document a race condition will not occur this is because the timer class has already a segment of the sequence number avoiding any other thread to access this segment, so we won't have a scenario where two threads are trying to access the same segment and there for no race conditions will occur. Also, the segment will not be retransmitted before it was initialized because stop and go happens one after another and with the segment within the Timer class we can make sure that segments are sent one after the other. Thank you for your time to read this. Hope you have a great day.