

Khrystian Clark

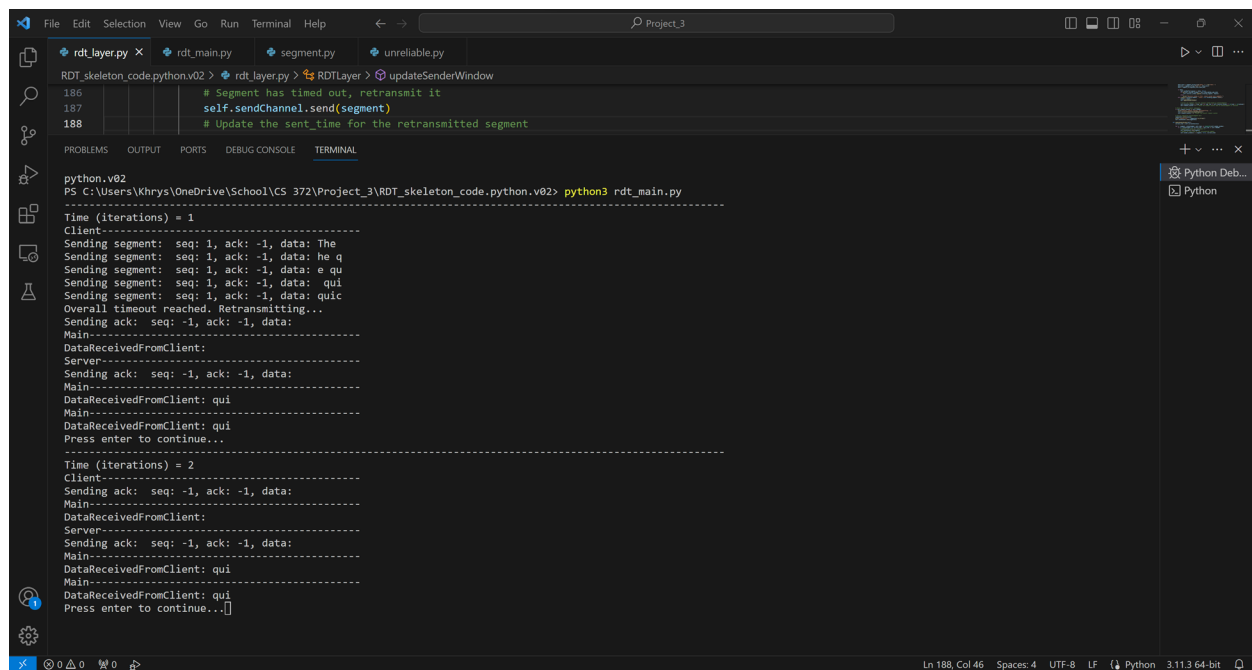
11/28/2023

- Introduction
 - Purpose of the Assignment: The purpose of this assignment is implement a reliable data transfer layer over an unreliable network.
 - Bulk of the logic: The bulk of the logic in this project lives in the `rdt_layer.py` file, in the `RDTPayer` class, in the functions for `processSend`, and `processReceiveAndSendResponse`. Like the logic for pipelining, cumulative acks, flow control, as well as initializing the Go-Back-N and cumulative ack logic.
 - Timeouts: I was not able to directly resolve timeouts, but if they were resolved, the program would finish and give the output window all of the needed information from the example image in the instructions.
 - Packet dropping: Packet dropping was partially handled when I figured out how to implement more clear logic on Go-Back-N. It was something I was stuck on that sort of made more sense as I went through the project rubric.
 - Retransmission: This policy was implemented by the addition of a helper function, `retransmitUnacknowledgedSegments`, which literally does exactly what is in its name. It retransmits for segments based on timeouts.
 - Scope of the assignment:
 1. Run the `rdt_main.py` that tests the logic of implemented to create a reliable data transfer in an unreliable environment.
 - Development Environment:
 1. Python3
 2. Written in python
 3. IDE's used to create/test: VS Code, GitHub CodeSpace, PowerShell
-
- Instructions
 - Note: These are python3 instructions
 1. Save and extract the RDT zip file locally
 2. In the terminal, map to the location of the saved `segment.py`, `unreliable.py`, `rdt_main.py` and `rdt_layer.py` files
 3. Open the `rdt_main.py` file and uncomment the line of code in the `__main__` that you want to run the segmenting for
 - Just one of them can be uncommented at a time
 4. In the terminal, run the program `"python3 rdt_main"`
 5. Review the output in the terminal response window.
- Screenshot of running locally

```
Administrator: PowerShell
PS C:\Users\Khrys\OneDrive\School\CS 372\Project_3\RDt_skeleton_code.python.v02> python3 rdt_main.py

Time (iterations) = 1
Client-----
Sending segment: seq: 1, ack: -1, data: The
Sending segment: seq: 1, ack: -1, data: he q
Sending segment: seq: 1, ack: -1, data: e qu
Sending segment: seq: 1, ack: -1, data: qui
Sending segment: seq: 1, ack: -1, data: quic
Overall timeout reached. Retransmitting...
Sending ack: seq: -1, ack: -1, data:
Main-----
DataReceivedFromClient:
Server-----
Sending ack: seq: -1, ack: -1, data:
Main-----
DataReceivedFromClient: qui
Main-----
DataReceivedFromClient: qui
Press enter to continue...

Time (iterations) = 2
Client-----
Sending ack: seq: -1, ack: -1, data:
Main-----
DataReceivedFromClient:
Server-----
Sending ack: seq: -1, ack: -1, data:
Main-----
DataReceivedFromClient: qui
Main-----
DataReceivedFromClient: qui
Press enter to continue...
```



```
File Edit Selection View Go Run Terminal Help
Project_3
rdt_layer.py rdt_main.py segment.py unreliable.py
RDt_skeleton_code.python.v02 > rdt_layer.py > RDTLayer > updateSenderWindow
186 # Segment has timed out, retransmit it
187 self.sendChannel.send(segment)
188 # Update the sent_time for the retransmitted segment

PROBLEMS OUTPUT PORTS DEBUG CONSOLE TERMINAL
python.v02
PS C:\Users\Khrys\OneDrive\School\CS 372\Project_3\RDt_skeleton_code.python.v02> python3 rdt_main.py

Time (iterations) = 1
Client-----
Sending segment: seq: 1, ack: -1, data: The
Sending segment: seq: 1, ack: -1, data: he q
Sending segment: seq: 1, ack: -1, data: e qu
Sending segment: seq: 1, ack: -1, data: qui
Sending segment: seq: 1, ack: -1, data: quic
Overall timeout reached. Retransmitting...
Sending ack: seq: -1, ack: -1, data:
Main-----
DataReceivedFromClient:
Server-----
Sending ack: seq: -1, ack: -1, data:
Main-----
DataReceivedFromClient: qui
Main-----
DataReceivedFromClient: qui
Press enter to continue...

Time (iterations) = 2
Client-----
Sending ack: seq: -1, ack: -1, data:
Main-----
DataReceivedFromClient:
Server-----
Sending ack: seq: -1, ack: -1, data:
Main-----
DataReceivedFromClient: qui
Main-----
DataReceivedFromClient: qui
Press enter to continue...]
```

- Include comments / questions (optional)
 - Getting the Go-Back_N logic to make sense was the most challenging portion of this assignment.
 - I spent a lot of my time going back to adjust small portions to make sure the send and receives handled/made sense in comparison with the textbook logic from ch 3.4-3.5

- This was a very fun and challenging project. I love the troubleshooting aspects of the projects in this course that make me re-think but with more and more logic added.