PHASE 3 REPORT
EECE 4830 Network Design – RDT 3.0 Over Unreliable UDP with Bit Errors and Loss

Team Members:
Jacob Alicea, Josiah Concepcion, Jamie Oliphant, Tim Saari

Course: EECE 4830 Spring 2025
Instructor: Dr. Vinod Vokkarane

Overview

This report documents the design, implementation, and testing of our RDT 3.0 protocol over an unreliable UDP channel. Building on our previous RDT 2.2 implementation, Phase 3 adds support for handling both bit-errors and packet losses by incorporating a countdown timer to detect lost packets quickly. We test five scenarios using a JPEG file (“tiger.jpg” , 1.1 MB) to evaluate performance under different conditions.

Design Details

Goal: Implement a reliable data transfer (RDT) protocol (RDT 3.0) over UDP that not only handles bit-errors (as in RDT 2.2) but also detects and recovers from both ACK and data packet losses.

Key Features:

- **Sequence Numbers & Packet Structure**:
  - Each packet includes a 4-byte sequence number (using struct.pack('!I', seq_num)) followed by 1024 bytes of file data.
- **Countdown Timer**:
  - A 50 ms timeout is used to detect lost packets quickly. If an ACK is not received before the timer expires, the sender retransmits the packet.
- **Error and Loss Simulation**:
  - Scenario 1 (No Loss/Bit-Errors): error_rate = 0.0 and loss_rate = 0.0.
  - Scenario 2 (ACK Packet Bit-Error): The sender simulates ACK corruption by flipping a bit in the ACK packet with a given error rate.
  - Scenario 3 (Data Packet Bit-Error): The receiver simulates data corruption by flipping a bit in the data payload with a given error rate.
  - Scenario 4 (ACK Packet Loss): The sender simulates ACK loss by dropping received ACK packets with a specified loss rate.

    - Scenario 5 (Data Packet Loss): The receiver simulates data loss by dropping incoming data packets with a specified loss rate.
- **Performance Logging**:
    - Each test run's completion time is recorded in "completion_times.csv" (the summarized data appears in sorted_completion_times.txt) to analyze the impact of increasing error/loss rates.

sender.py

1. **File Reading & Packet Construction**:
    - Reads "tiger.jpg" in 1024-byte chunks and prepends each chunk with a 4-byte sequence number.
2. **Packet Transmission**:
    - Sends each packet via a UDP socket to the receiver at 127.0.0.1 on port 5001.
3. **ACK Handling & Countdown Timer**:
    - Waiting for an ACK. If the ACK (after possible corruption simulation) matches the current sequence number, the sender flips the sequence number and proceeds.
    - If the correct ACK isn't received within 50 ms, the sender retransmits the packet.
4. **Error/Loss Simulation**:
    - The sender simulates ACK bit-errors (Scenario 2) and ACK packet loss (Scenario 4) using configurable error_rate and loss_rate parameters.

receiver.py

1. **Listening for Packets**:
    - Binds to UDP port 5001 and waits for incoming packets.
2. **Packet Processing**:
    - Extracts the 4-byte sequence number and the data payload.
3. **Data Corruption & Loss Simulation**:
    - Simulates data corruption (Scenario 3) by potentially flipping a bit in the first byte of the payload.
    - Simulates data packet loss (Scenario 5) by randomly dropping packets.
4. **ACK Handling & File Assembly**:
    - If the packet's sequence number matches the expected sequence it writes the data to "received_tiger.jpg" and flips the expected sequence number.
    - Sends an ACK (also subject to loss simulation for ACK loss) back to the sender.
    - An empty packet (only containing the sequence number) signals EOF.

1. **How to Run the Programs**
    - Place sender.py, receiver.py, tiger.jpg, in the same directory.
    - Open a terminal/IDE, navigate to the project directory, and run: python receiver.py
    - The receiver binds to UDP port 5001 and waits for incoming packets, writing data to “received_tiger.jpg”.
    - Open a second terminal in the same directory and run: python sender.py
    - The sender will read “tiger.jpg” and iterate through the five test scenarios (varying error/loss rates from 0% to 60% in 5% increments)..
    - Completion times for each test run are logged to “completion_times.csv”.

Testing Scenarios

1. **Scenario 1 – No Loss/Bit-Errors**:

    - Parameters: error_rate = 0.0, loss_rate = 0.0
    - Expected Outcome: Minimal retransmissions and fast transfer (completion times around 0.07 seconds).

2. **Scenario 2 – ACK Packet Bit-Error**:

    - Parameters: Varying error_rate for ACK corruption (loss_rate = 0.0).
    - Expected Outcome: Increased retransmissions due to corrupted ACKs; completion times gradually increase (e.g., from around 3 seconds to over 300 seconds at high error rates).

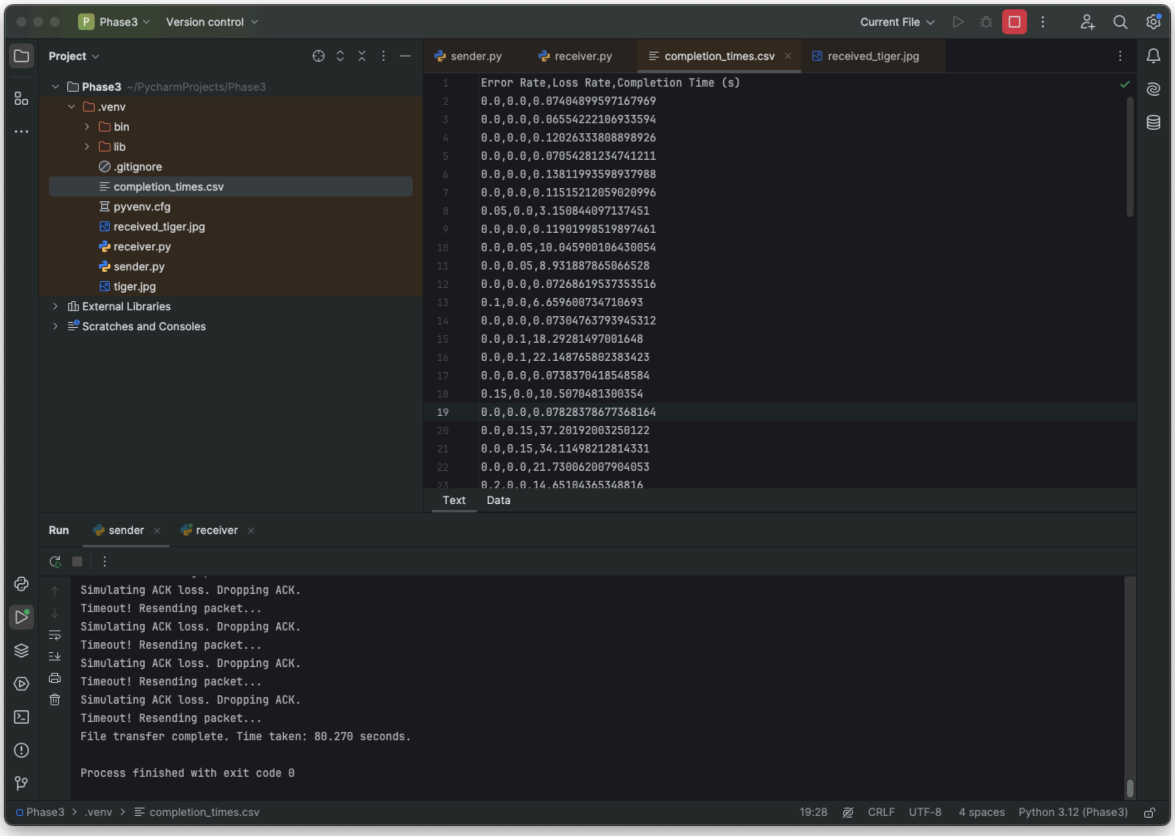
3. **Scenario 3 – Data Packet Bit-Error**:

    - Parameters: Varying error_rate for data corruption (loss_rate = 0.0).
    - Expected Outcome: The receiver detects corrupted data and discards packets, triggering retransmissions and longer transfer times.

4. **Scenario 4 – ACK Packet Loss**:

    - Parameters: Varying loss_rate for ACK loss (error_rate = 0.0).
    - Expected Outcome: Missing ACKs force the sender to timeout and retransmit, increasing overall completion time.

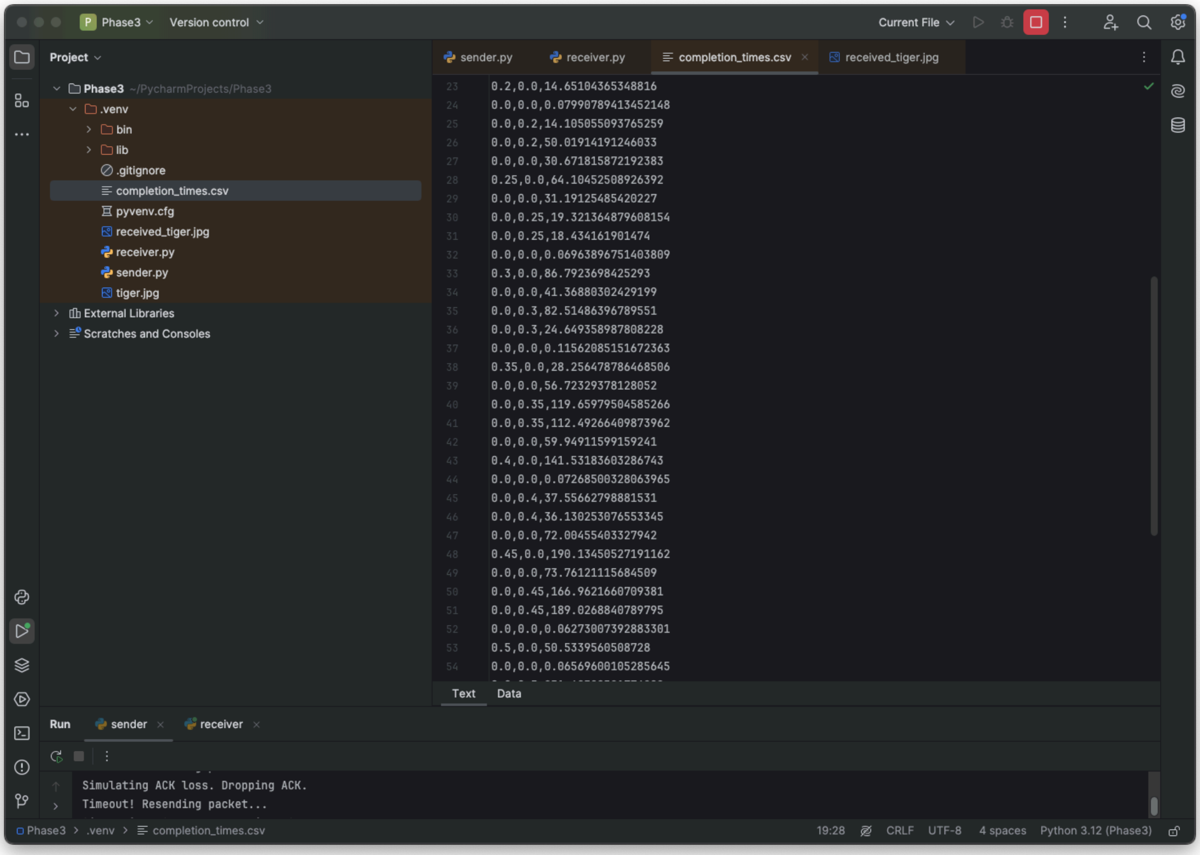
5. **Scenario 5 – Data Packet Loss**:
    - Parameters: Varying loss_rate for data packet loss (error_rate = 0.0).
    - Expected Outcome: Dropped data packets result in more retransmissions and significantly longer transfer times.



```
Error Rate,Loss Rate,Completion Time (s)
0.0,0.0,0.07404899597167969
0.0,0.0,0.06554222106933594
0.0,0.0,0.12026333808898926
0.0,0.0,0.07054281234741211
0.0,0.0,0.13811993598937988
0.0,0.0,0.11515212059020996
0.05,0.0,3.150844097137451
0.0,0.0,0.11901998519897461
0.0,0.05,10.045900106430054
0.0,0.05,8.931887865066528
0.0,0.0,0.07268619537353516
0.1,0.0,6.659600734710693
0.0,0.0,0.07304763793945312
0.0,0.1,18.29281497001648
0.0,0.1,22.148765802383423
0.0,0.0,0.0738370418548584
0.15,0.0,10.5070481300354
0.0,0.0,0.07828378677368164
0.0,0.15,37.20192003250122
0.0,0.15,34.11498212814331
0.0,0.0,21.730062007904053
0.2,0.0,14.65104365348816
```

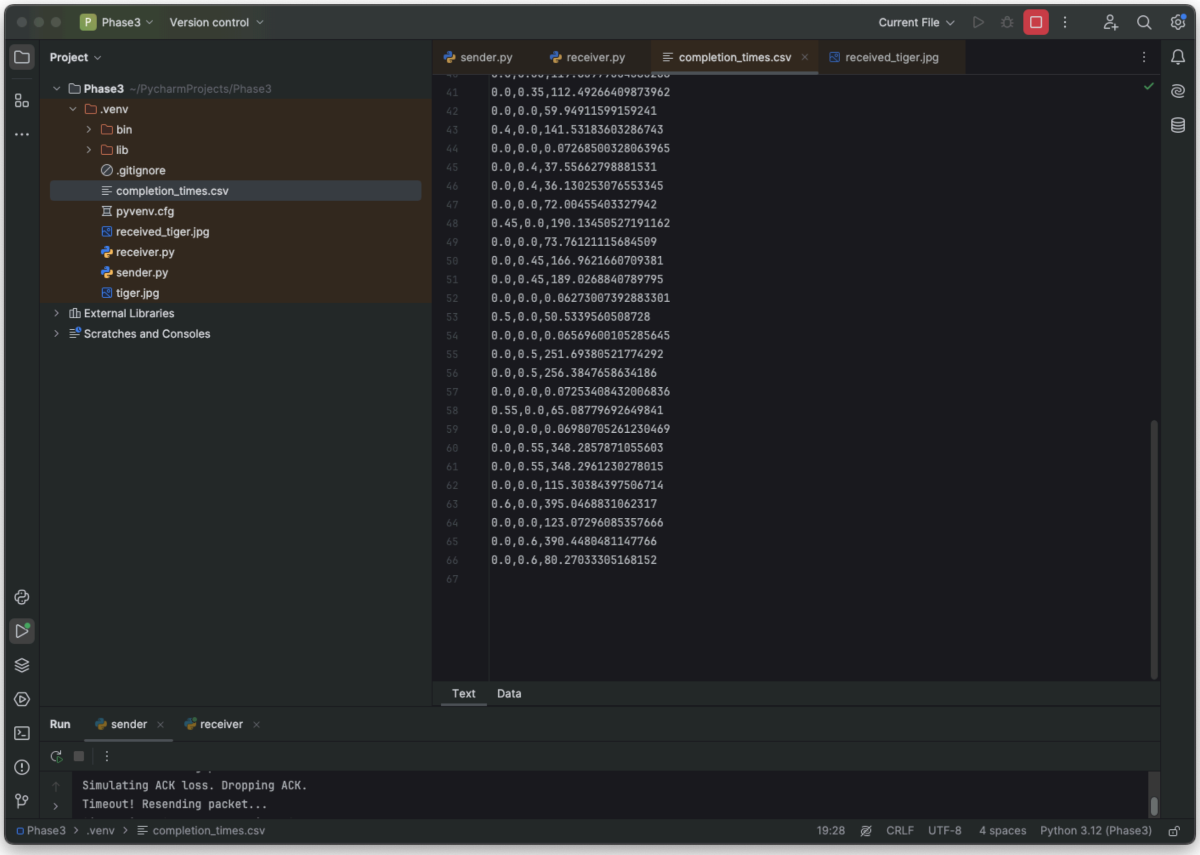
```
Simulating ACK loss. Dropping ACK.
Timeout! Resending packet...
Simulating ACK loss. Dropping ACK.
Timeout! Resending packet...
Simulating ACK loss. Dropping ACK.
Timeout! Resending packet...
Simulating ACK loss. Dropping ACK.
Timeout! Resending packet...
File transfer complete. Time taken: 80.270 seconds.

Process finished with exit code 0
```



```
0.2,0.0,14.65104365348816
0.0,0.0,0.07990789413452148
0.0,0.2,14.105055093765259
0.0,0.2,50.01914191246033
0.0,0.0,30.671815872192383
0.25,0.0,64.10452508926392
0.0,0.0,31.19125485420227
0.0,0.25,19.321364879608154
0.0,0.25,18.434161901474
0.0,0.0,0.06963896751403809
0.3,0.0,86.7923698425293
0.0,0.0,41.36880302429199
0.0,0.3,82.51486396789551
0.0,0.3,24.649358987808228
0.0,0.0,0.11562085151672363
0.35,0.0,28.256478786468506
0.0,0.0,56.72329378128052
0.0,0.35,119.65979504585266
0.0,0.35,112.49266409873962
0.0,0.0,59.94911599159241
0.4,0.0,141.53183603286743
0.0,0.0,0.07268500328063965
0.0,0.4,37.55662798881531
0.0,0.4,36.130253076553345
0.0,0.0,72.00455403327942
0.45,0.0,190.13450527191162
0.0,0.0,73.76121115684509
0.0,0.45,166.9621660709381
0.0,0.45,189.0268840789795
0.0,0.0,0.06273007392883301
0.5,0.0,50.5339560508728
0.0,0.0,0.06569600105285645
```

```
Simulating ACK loss. Dropping ACK.
Timeout! Resending packet...
```



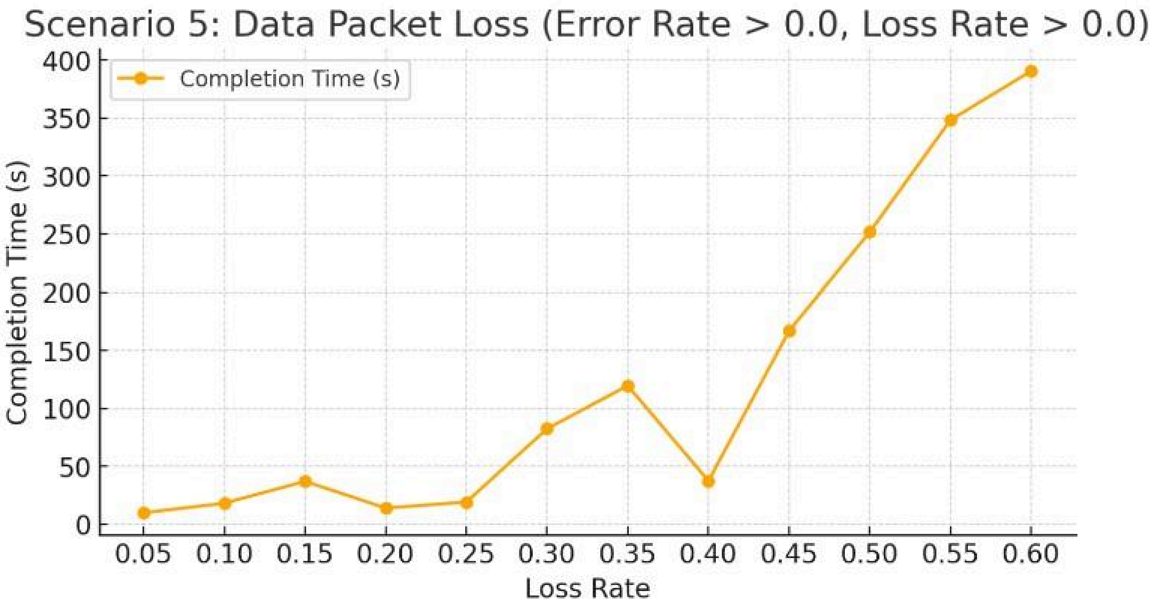
```
0.0,0.35,112.49266409873962
0.0,0.0,59.94911599159241
0.4,0.0,141.53183603286743
0.0,0.0,0.07268500328063965
0.0,0.4,37.55662798881531
0.0,0.4,36.130253076553345
0.0,0.0,72.00455403327942
0.45,0.0,190.13450527191162
0.0,0.0,73.76121115684509
0.0,0.45,166.9621660709381
0.0,0.45,189.0268840789795
0.0,0.0,0.06273007392883301
0.5,0.0,50.5339560508728
0.0,0.0,0.06569600105285645
0.0,0.5,251.69380521774292
0.0,0.5,256.3847658634186
0.0,0.0,0.07253408432006836
0.55,0.0,65.08779692649841
0.0,0.0,0.06980705261230469
0.0,0.55,348.2857871055603
0.0,0.55,348.2961230278015
0.0,0.0,115.30384397506714
0.6,0.0,395.0468831062317
0.0,0.0,123.07296085357666
0.0,0.6,390.4480481147766
0.0,0.6,80.27033305168152
```

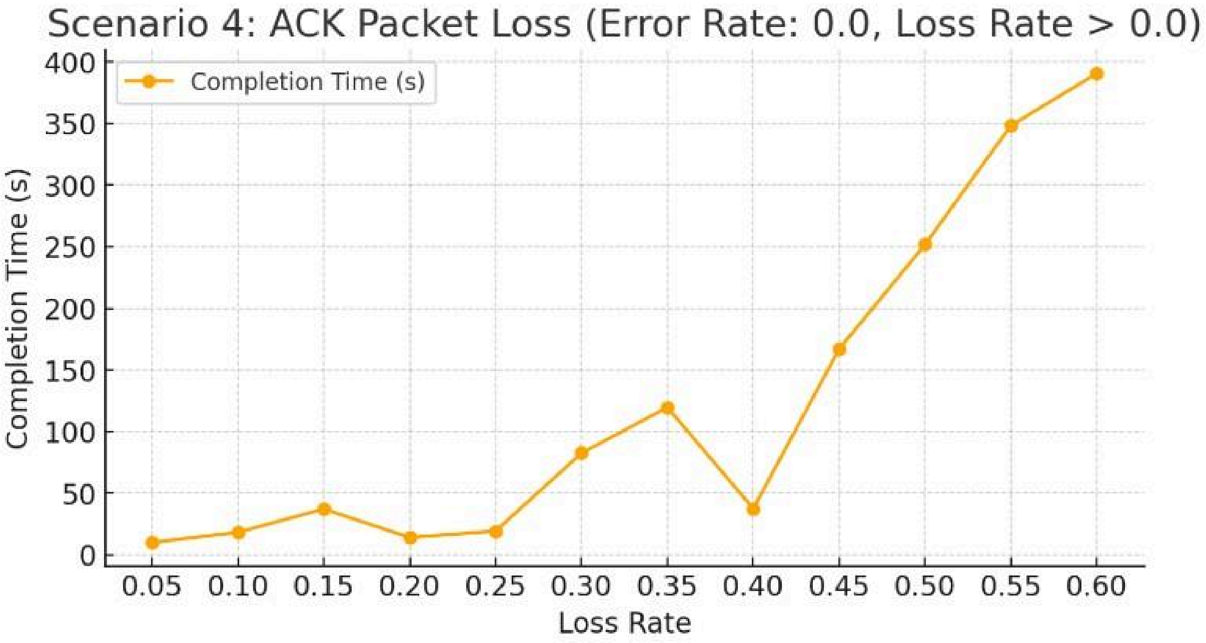
```
Simulating ACK loss. Dropping ACK.
Timeout! Resending packet...
```

Performance Analysis

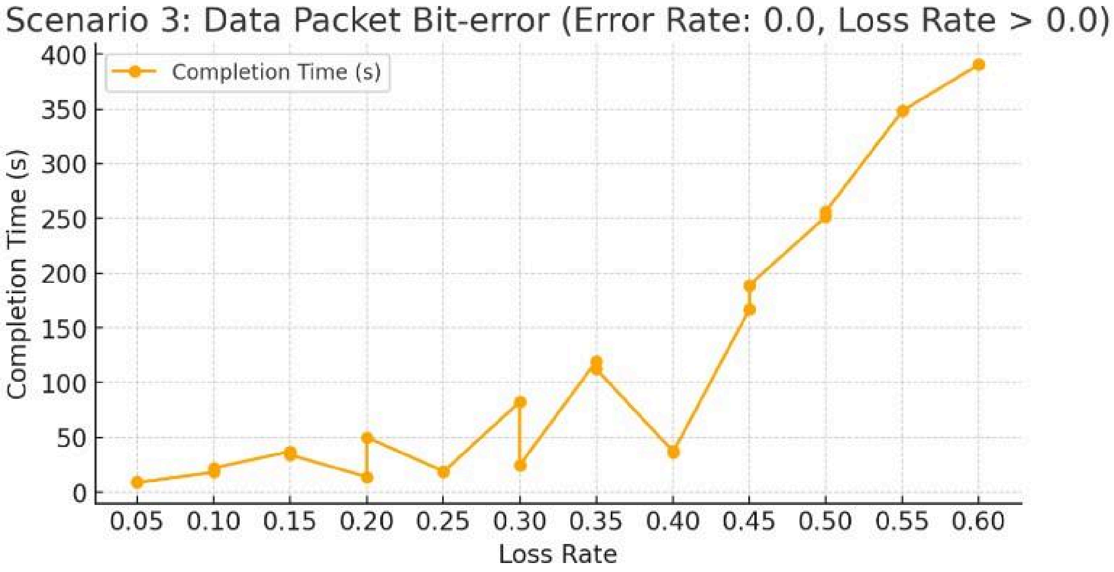
Using our recorded data in sorted_completion_times.txt we observe the following:

> **Scenario 1** shows extremely low completion times averaging around 0.07 seconds under error-free conditions.
>
> **Scenarios 2–5** demonstrate a clear increase in transfer time with higher error or loss rates. For example, in Scenario 2, as the error rate increases from 5% to 60%, completion times can range from a few seconds up to 400 seconds.
>
> These trends show the significant impact that packet corruption and loss have on throughput and overall transfer efficiency.

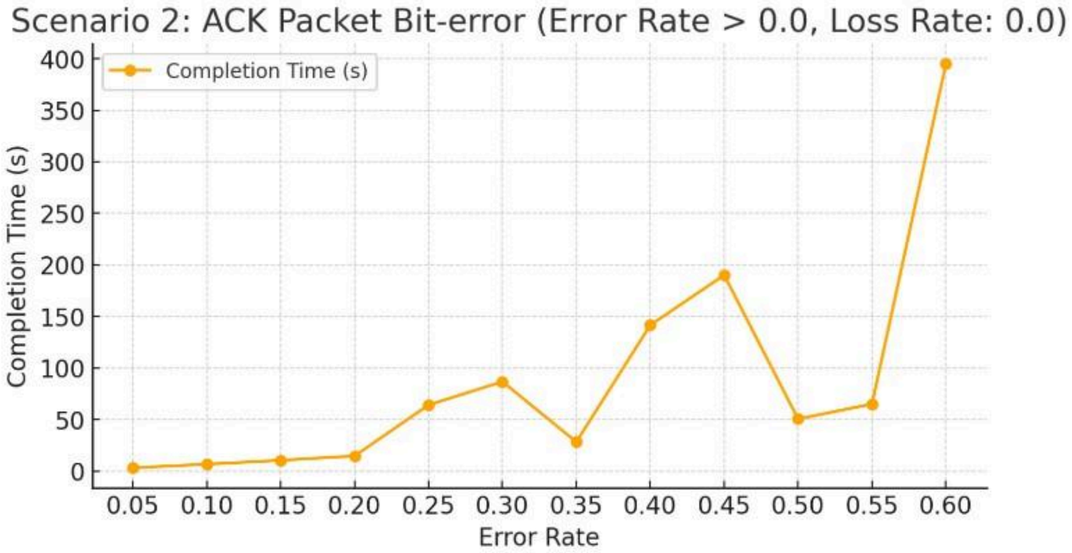

Scenario 5: Data Packet Loss (Error Rate > 0.0, Loss Rate > 0.0)
Completion Time (s)
Completion Time (s)
400
350
300
250
200
150
100
50
0
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
0.45
0.50
0.55
0.60
Loss Rate

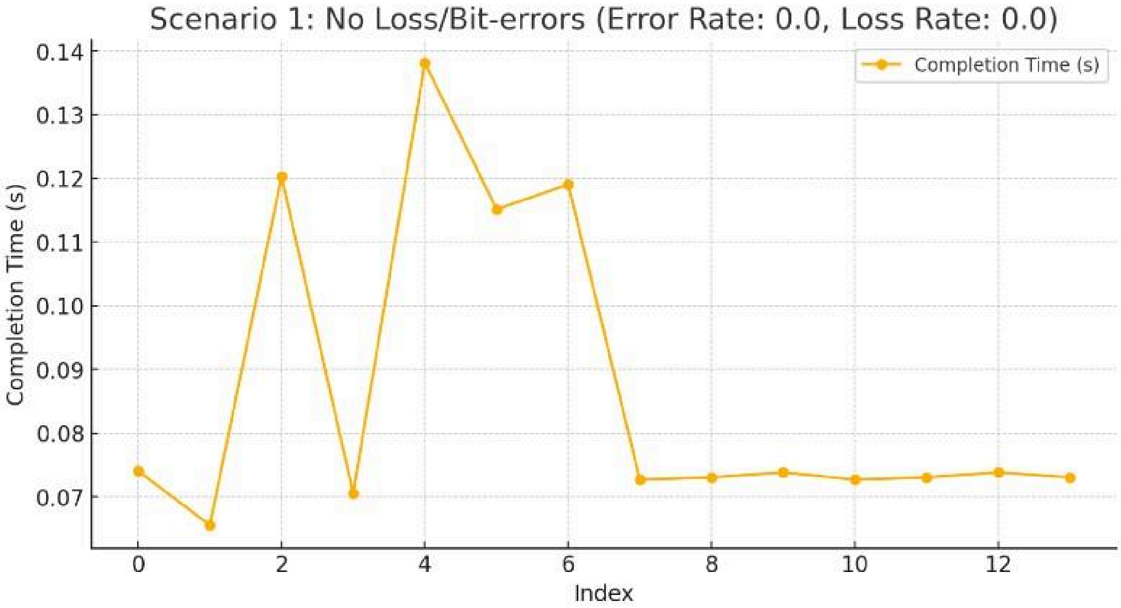


Scenario 4: ACK Packet Loss (Error Rate: 0.0, Loss Rate > 0.0)



Scenario 3: Data Packet Bit-error (Error Rate: 0.0, Loss Rate > 0.0)





Conclusion

- We successfully extended our reliable data transfer protocol to RDT 3.0, incorporating mechanisms to handle both bit-errors and packet losses.
- The introduction of a countdown timer is crucial in detecting lost packets promptly.
- Testing shows that even moderate levels of error or loss can lead to huge retransmission overhead which significantly increases file transfer times.
- The balance between ensuring data integrity and maintaining efficient transfer times is a constant battle