Ahmed Emad Barakat.
2807

Reliable Data Transfer

Stop-and-Wait, Selective-Repeat and Go-Back-N

Reliable Data Transfer

Stop-and-Wait, Selective-Repeat and Go-Back-N

✓ Packet Types:

1. *Signal Packet:*

- Length 2 Bytes short (header)
  - (Packet Length).
- Packet Type 2 Bytes short (header)
  - (Packet Type Enum Value).
- Check Sum 8 Bytes long (header)
  - (Internet CheckSum of packet data).
- Name ? Bytes String as byte[] (data)
  - (Name of Client).

Packet = header + options (int[]) + data = 12 Bytes + 0 + ?? Bytes.

2. *File Request Packet:*

- Length 2 Bytes short (header)
  - (Packet Length).
- Packet Type 2 Bytes short (header)
  - (Packet Type Enum Value).
- Check Sum 8 Bytes long (header)
  - (Internet CheckSum of packet data).
- File Name ? Bytes String as byte[] (data)
  - (Request File Name).

Packet = header + options (int[]) + data = 12 Bytes + 0 Bytes + ?? Bytes.

3. *Transfer Initialization Packet:*

- Length 2 Bytes short (header)
  - (Packet Length).
- Packet Type 2 Bytes short (header)
  - (Packet Type Enum Value).
- Check Sum 8 Bytes long (header)
  - (Internet CheckSum of packet data).
- File Size 4 Bytes int (options)
  - (Over all Data Size).
- Packet Size 4 Bytes int (options)
  - (Maximum Packet Size to receive "All except last").
- Window Size 4 Bytes int (options)
  - (According to Mode Enum The Window Size may be sent).

Packet = header + options (int[]) + data = 12 Bytes + 12 Bytes + 0 Bytes.

4. *Data Packet:*

- Length 2 Bytes Short (header)
  - (Packet Length).
- Packet Type 2 Bytes Short (header)
  - (Packet Type Enum Value).
- Check Sum 8 Bytes long (header)
  - (Internet CheckSum of packet data).
- Packet Sequence 4 Bytes int (options)
  - (Sequence Number of Packet).
- Data ? Bytes byte[] (data)
  - (Actual Data).

Packet = header + options (int[]) + data = 12 Bytes + 4 Bytes + ?? Bytes.

5. *Acknowledgment "AKN" Packet:*

- Length 2 Bytes Short (header)
  - (Packet Length).
- Packet Type 2 Bytes Short (header)
  - (Packet Type Enum Value).
- Check Sum 8 Bytes long (header)
  - (Internet CheckSum of packet data).
- AKN Number 4 Bytes int (options)
  - (Sequence Number of Successfully received Data Packet).

Packet = header + options (int[]) + data = 12 Bytes + 4 Bytes + ?? Bytes.

✓ Scenarios:

❖ Initialization for All Modes:

✳ ***Client***: Sends a signal packet with his name to the server and waits for signal packet response containing his name.

✳ ***Server***: Waits for any packet. When receiving a one, if the sender "client" has a Responder assigned to him then passes the message to him. If not, create a Responder for this client and then passes the packet to him.

(So any server after that means the responder for the client)

✳ ***Server***: Reads the signal packet and responds to the client with a signal packet containing his name.

✳ ***Client***: When receiving a signal packet contains his name, client sends a request file packet to server then waits for a transfer initialization packet.

✳ ***Server***: When receiving a request file packet , responder reads the file and initializes the transfer packets then sends a transfer initialization packet to the client.

✳ ***Client***: When receiving a transfer initialization packet, client initializes variables then waits for first data packet to be sent and the errors are handled according to the mode enum.

❖ Stop and Wait Mode:

✳ ***Server***: Start by sending data packet number 0 “0 indexing”.

✳ ***Server***: Set a timer for this packet and when the timer is trigged, server checks if an acknowledgment packet with this packet sequence number is received or not. If received then do nothing, otherwise send this data packet again.

✳ ***Client***: Waits for any data packet in order and when receiving one not in order, the client ignores it. Otherwise, client sends an acknowledgment packet with the received packet sequence number to the server.

✳ ***Server***: When receiving an acknowledgment packet stops the timers with the associated acknowledgment number and sends the next data packet.

❖ Selective Repeat Mode:

✳ ***Server***: Initializes window with window size.

✳ ***Client***: Initializes window with window size.

✳ ***Server***: Start by sending all packets in the window.

✳ ***Server***: Set a timer for each packet and when a timer is trigged, server checks if an acknowledgment packet with timer associated packet sequence number is received or not. If received then do nothing, otherwise send the data packet again.

✳ ***Client***: Waits for any data packet not required in order but required in the window and when receiving a one not in the window, the client ignores it. Otherwise, client sends an acknowledgment packet with the packet sequence number received to the server.

✳ ***Client***: If the received data packet is the first packet in the window, then move the window to the first unreceived data packet.

✳ ***Server***: When receiving an acknowledgment packet stop the timers with the associated acknowledgment number.

✳ ***Server***: If the received acknowledgment packet is the first packet in the window, then move the window to the first unreceived acknowledgment and send all packets not send in the window to the client.

❖ Go Back N Mode:

✳ ***Server***: Initializes window with window size.

✳ ***Server***: Start by sending all packets in window.

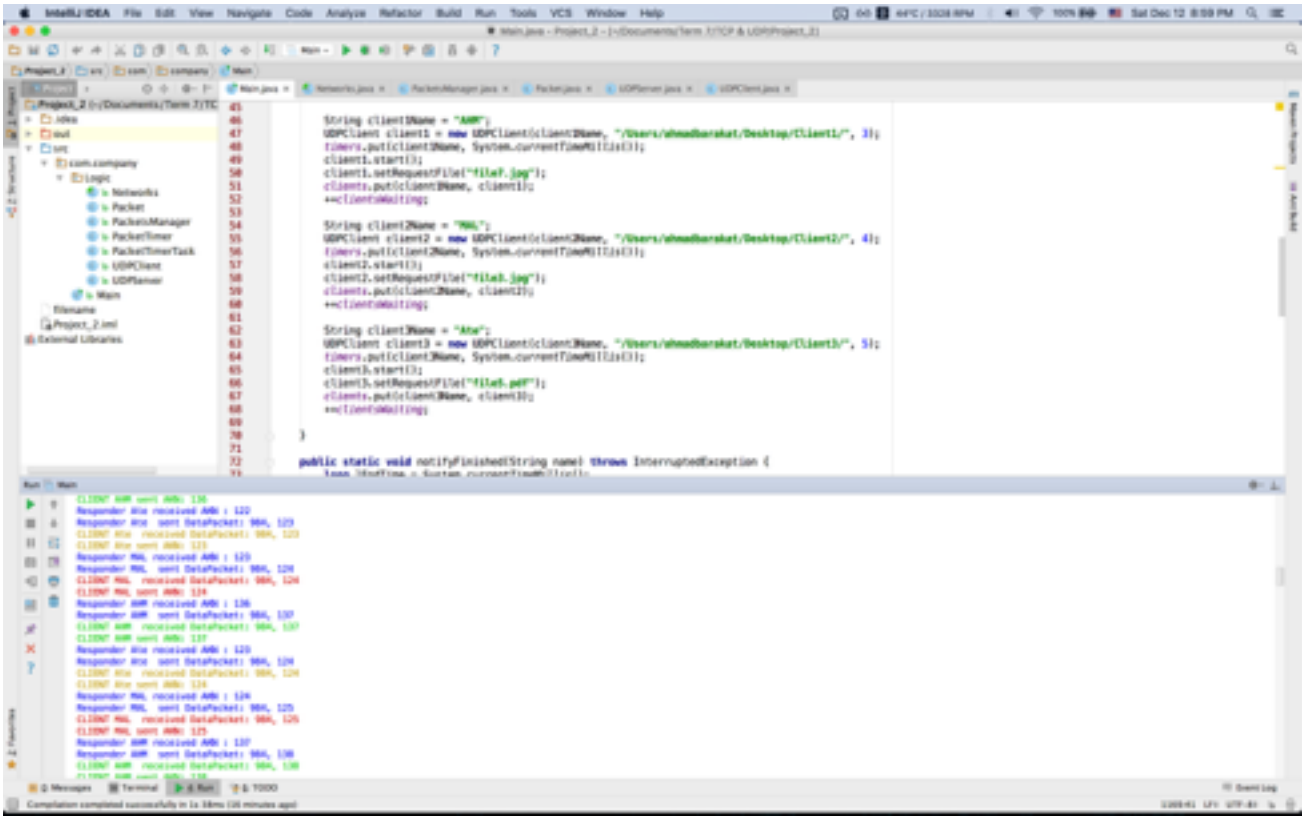
✳ ***Server***: Set a timer for the current window and when timer is trigged, server check if the first packet in the window is received or not , if true then do nothing. Otherwise send all the packets in the window again.

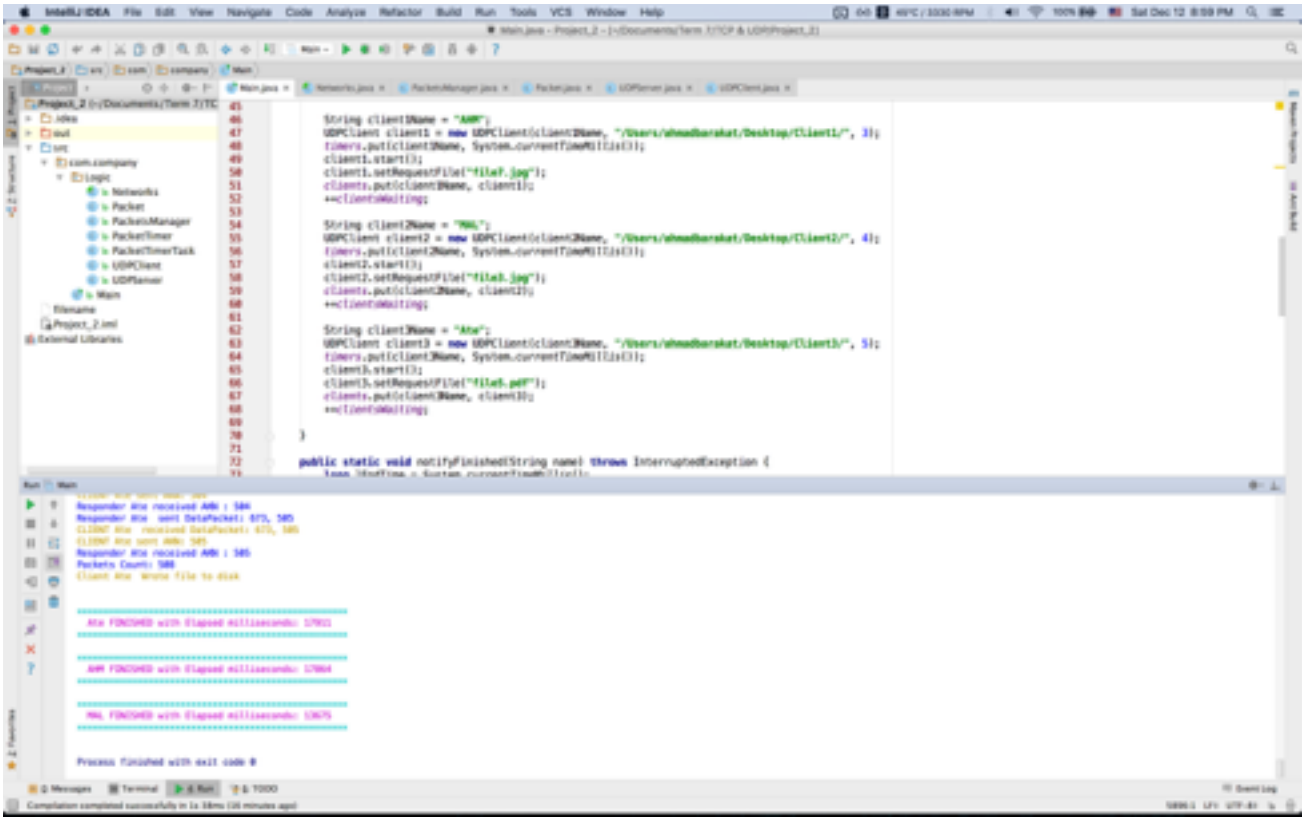
✳ ***Client***: Waits for any data packet in order and when receiving one not in order, the client ignores it. Otherwise, client sends an acknowledgment packet with the received packet sequence number to the server.

✳ ***Server***: When receiving an acknowledgment packet higher or equal than the window low, stop the timers with the associated window number.

✳ ***Server***: the received acknowledgment packet must be the first packet in the window or higher,. If so move the window to the first unreceived acknowledgment and send all packets not send in the window and assign a new timer for the new window.

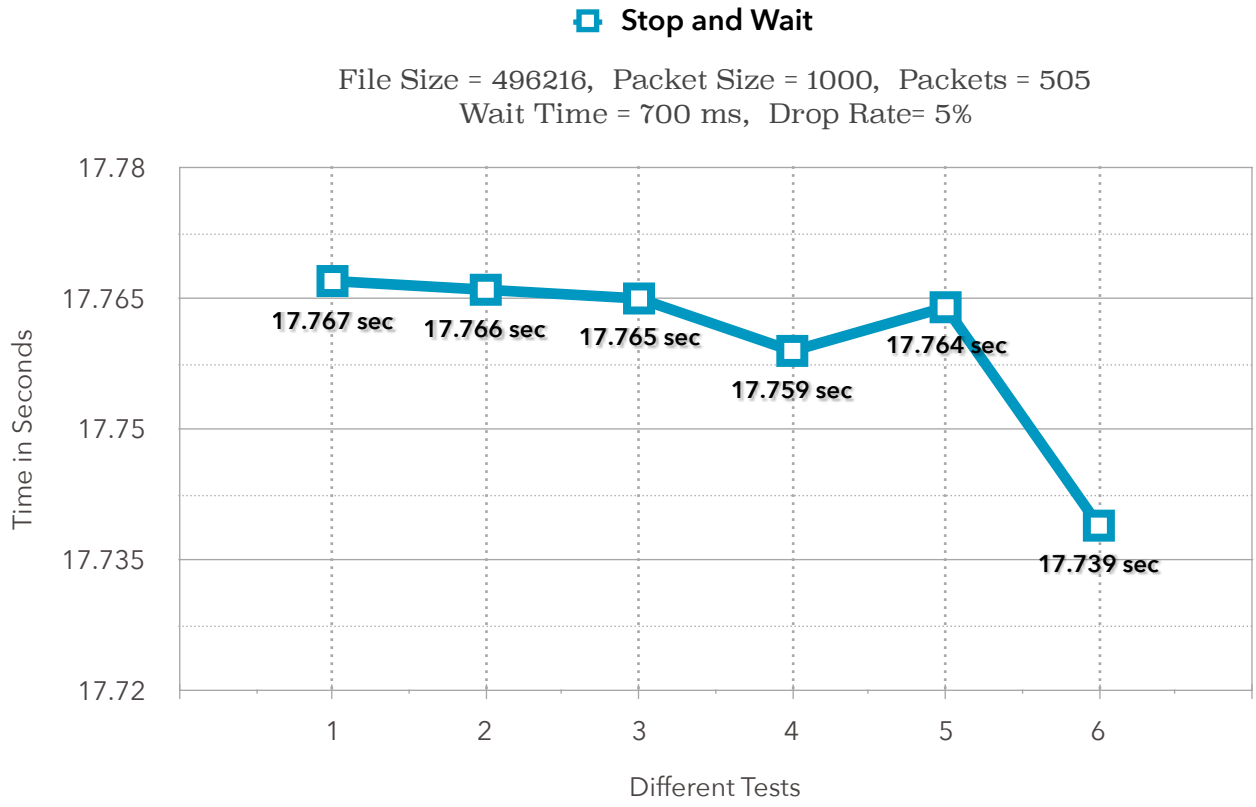
✓ Screenshots:

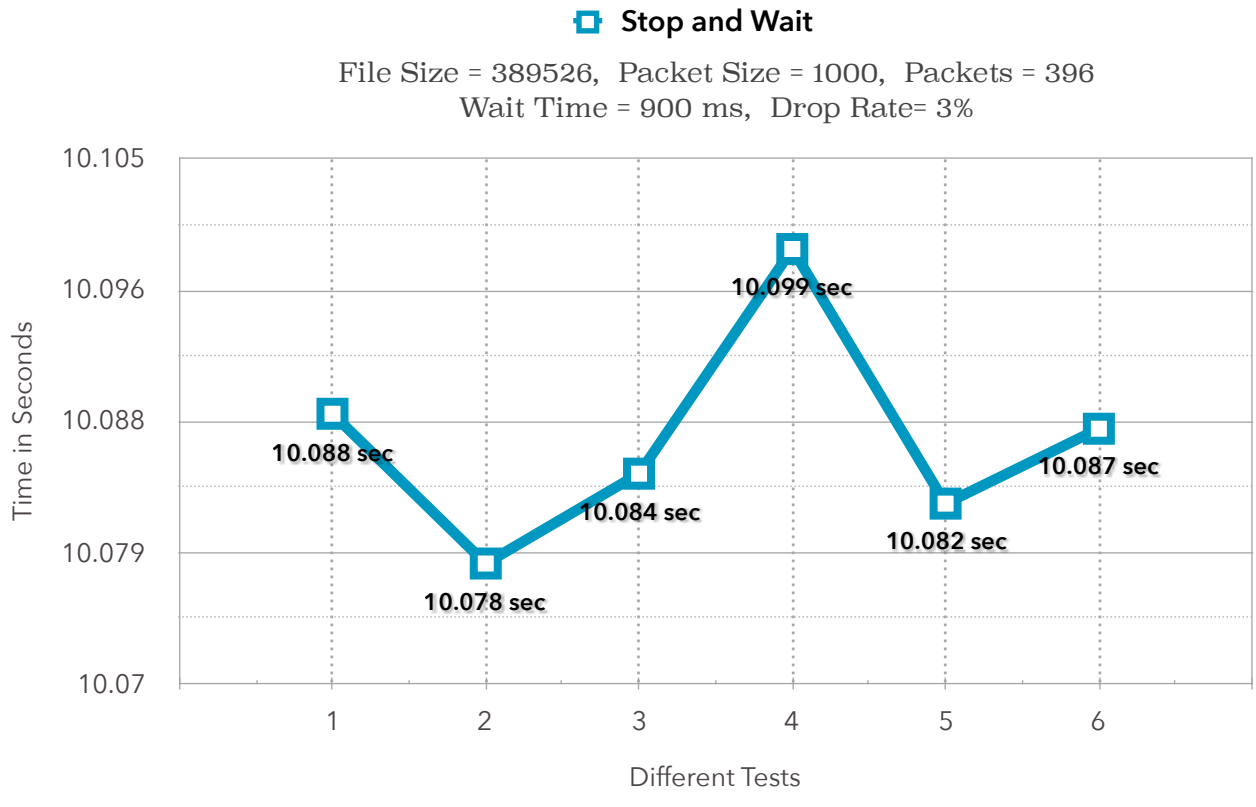


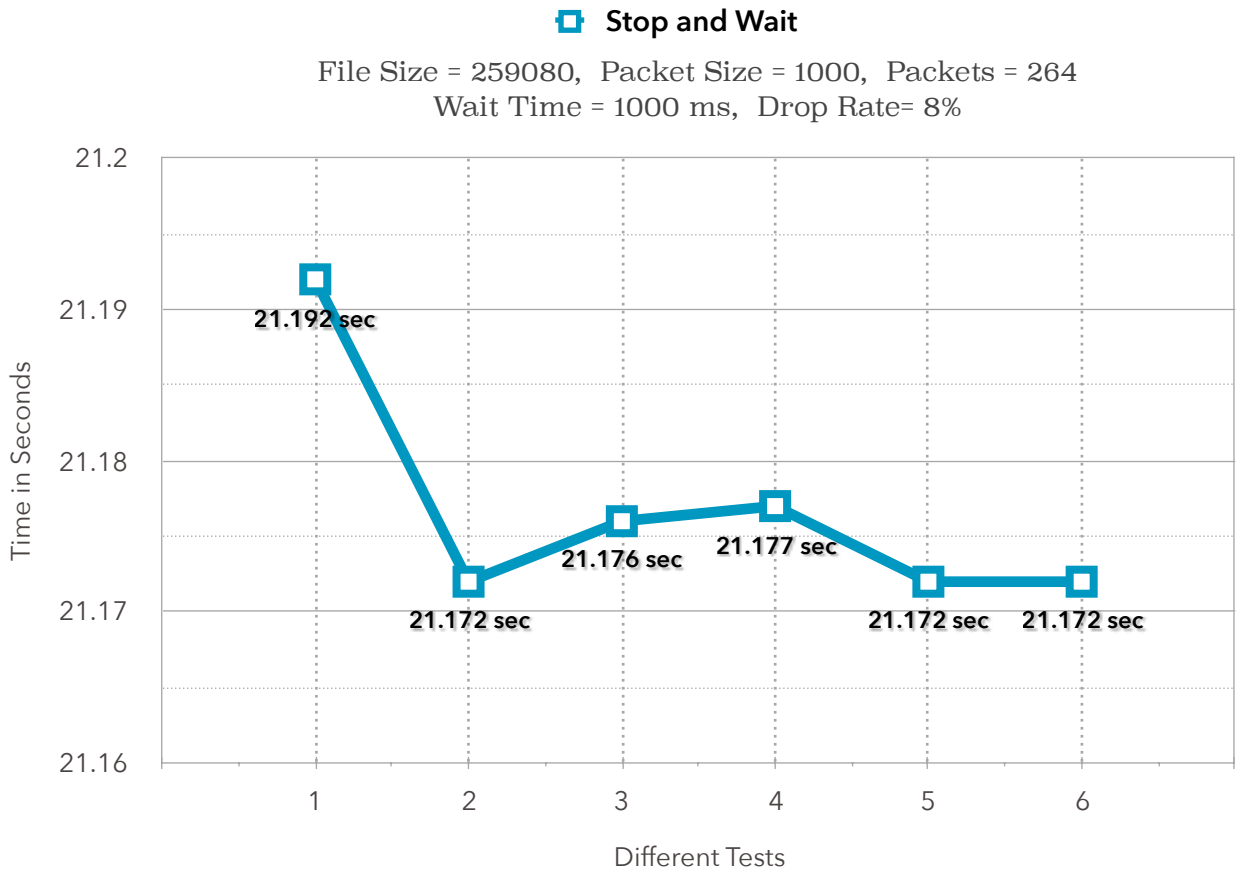


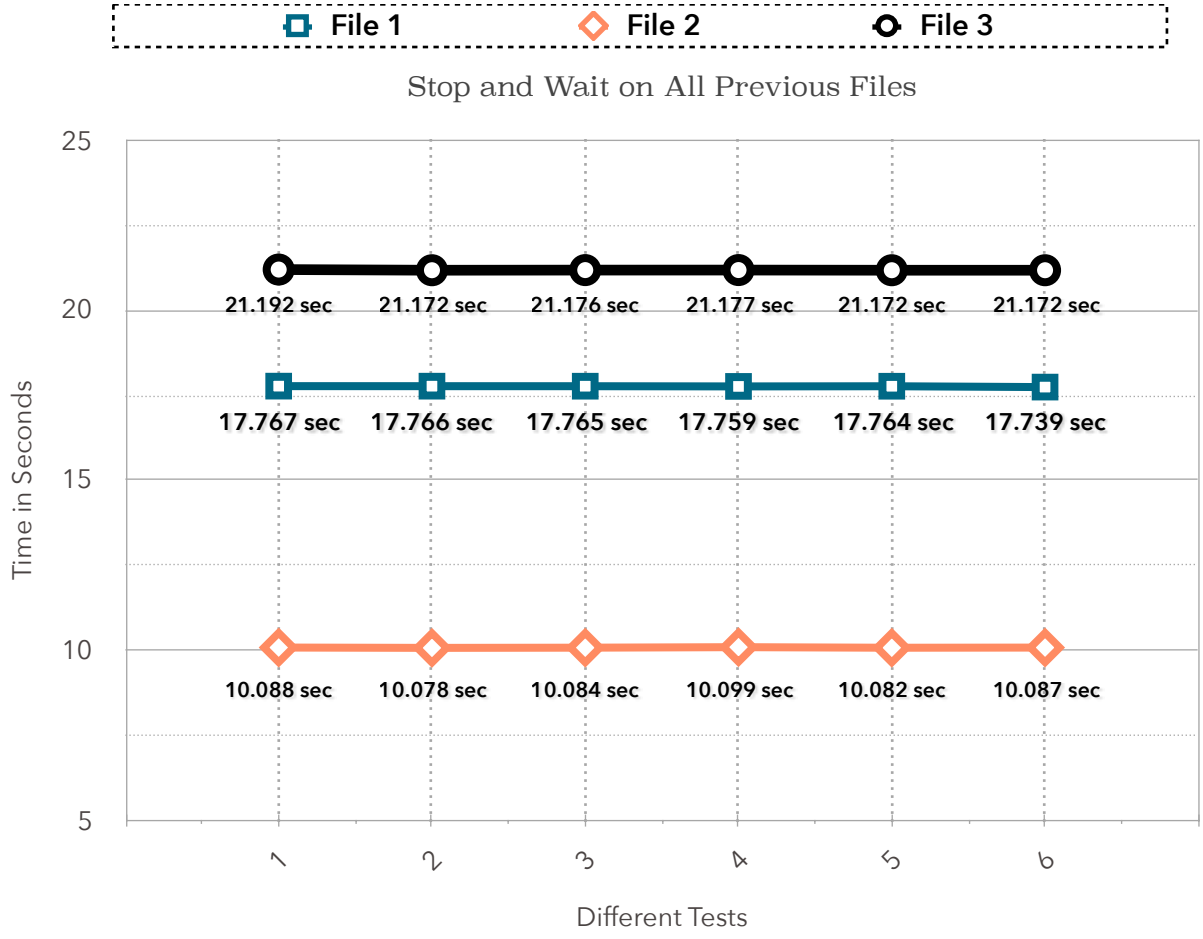
✓ Compares and Analysis:

✤ Stop and Wait:

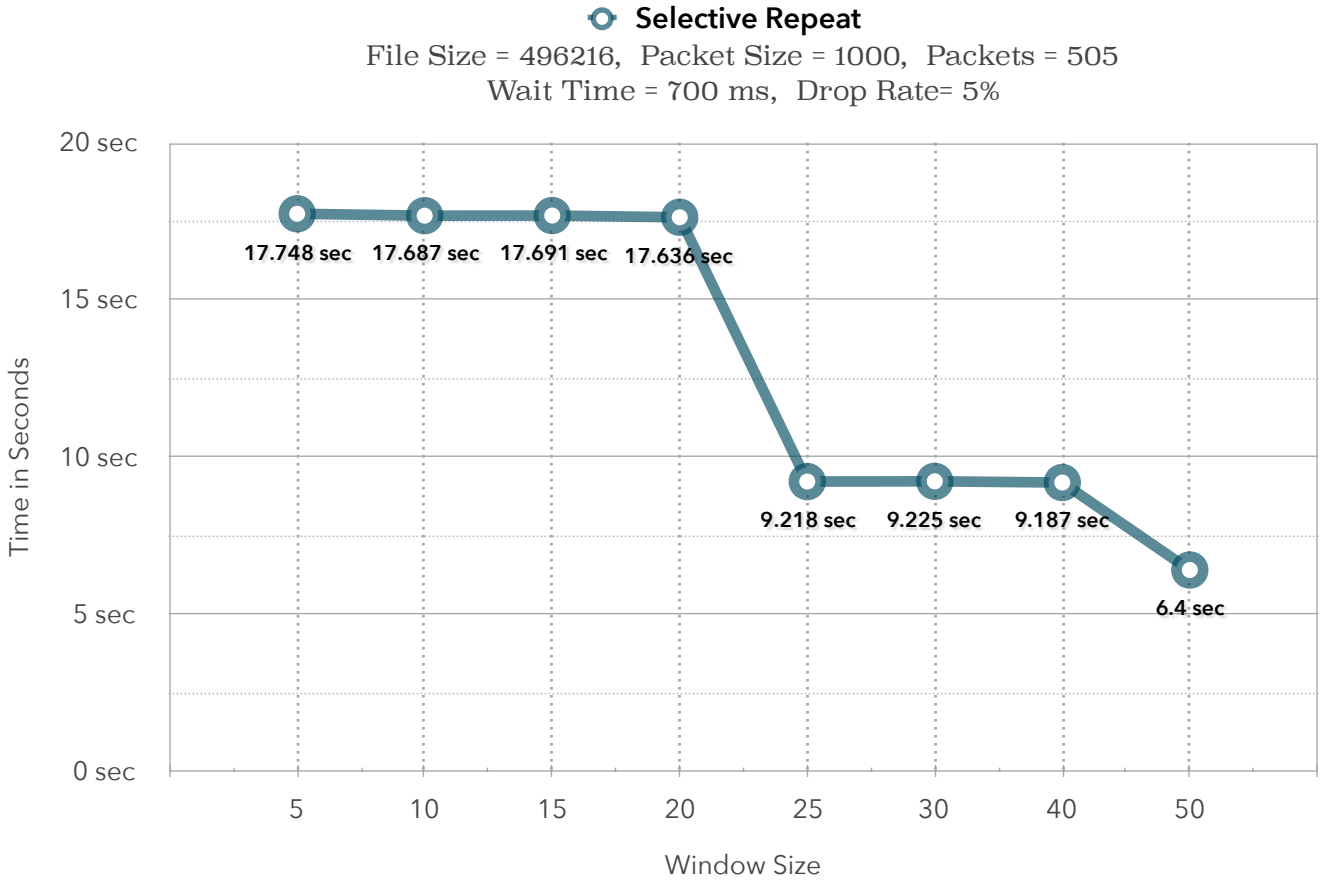


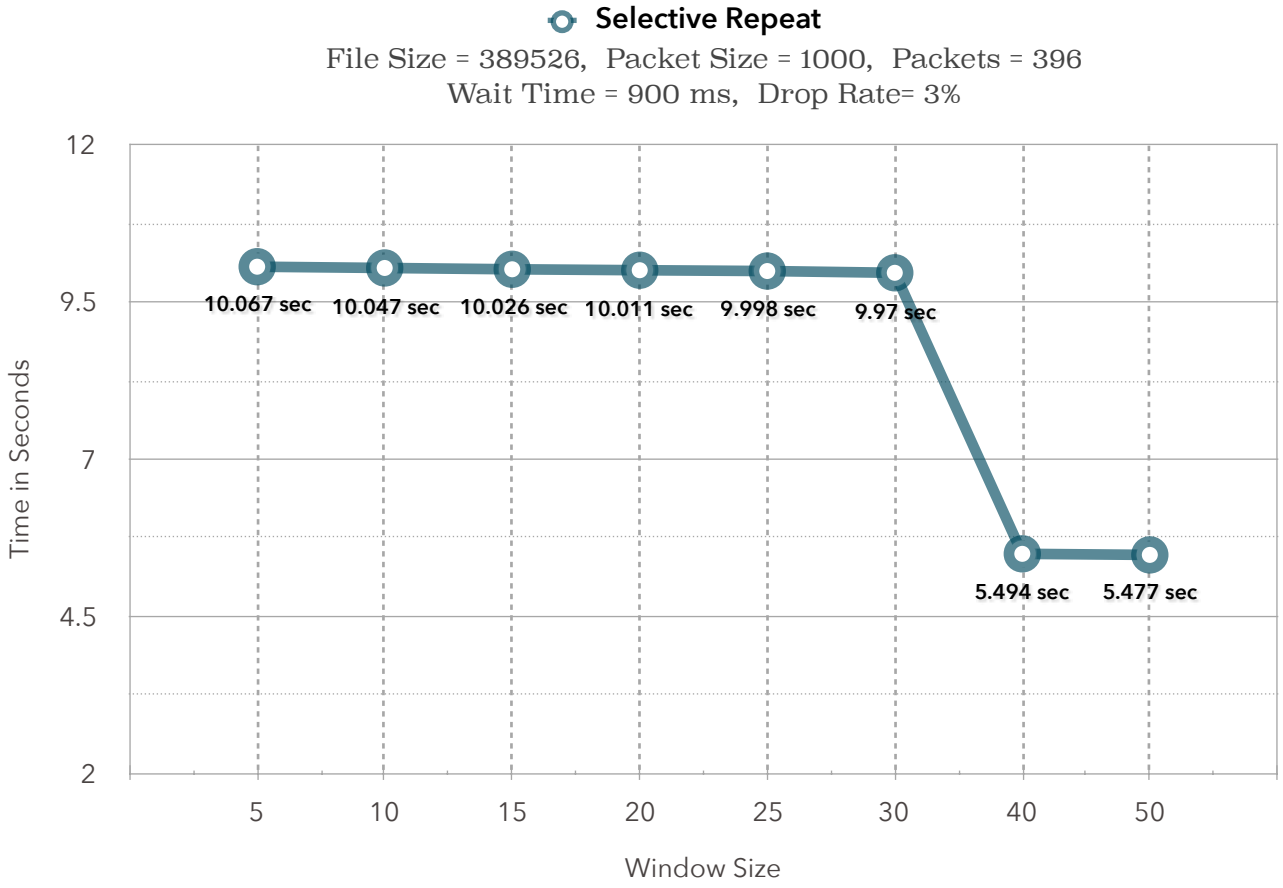


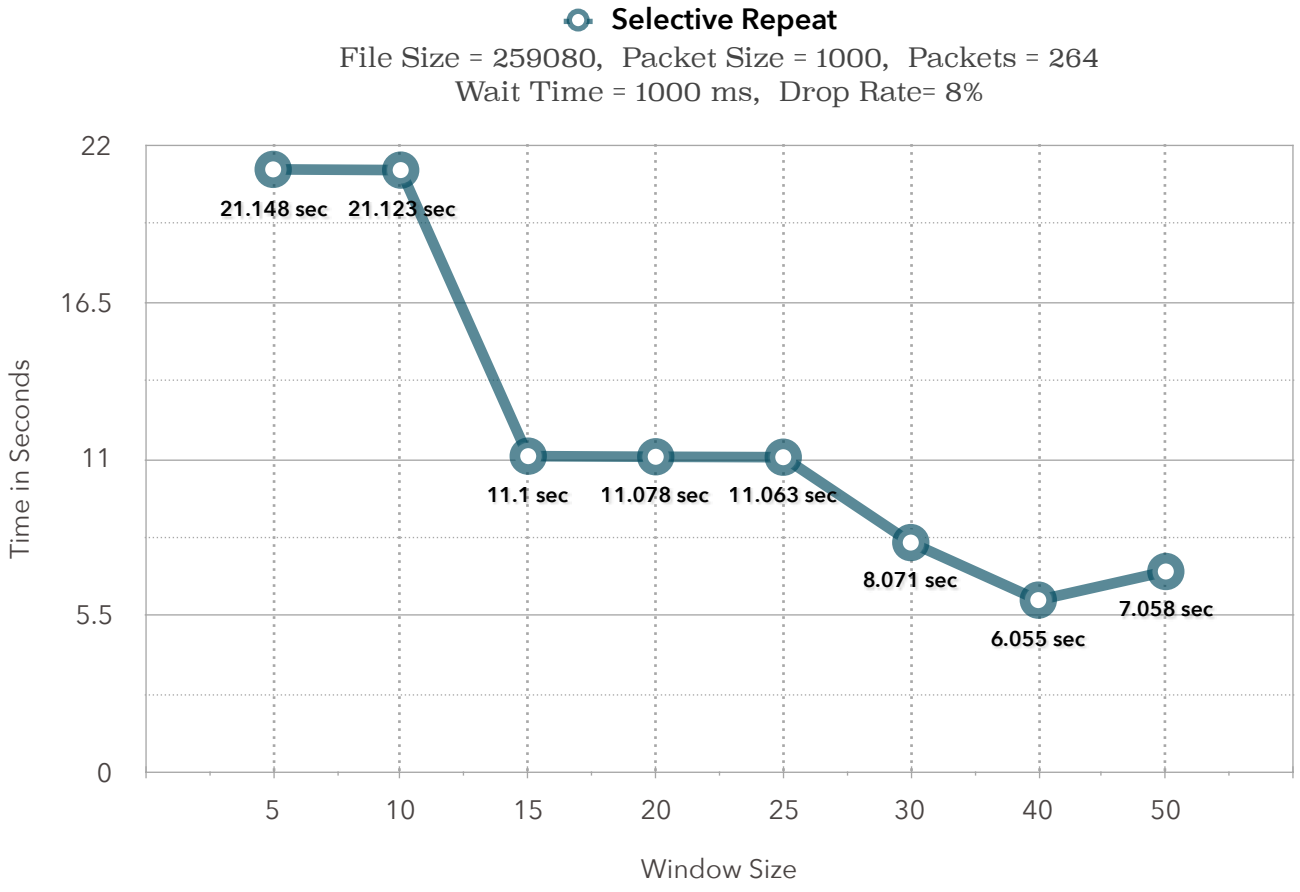

Stop and Wait
File Size = 259080, Packet Size = 1000, Packets = 264
Wait Time = 1000 ms, Drop Rate= 8%
21.2
21.19
21.18
21.17
21.16
Time in Seconds
21.192 sec
21.172 sec
21.176 sec
21.177 sec
21.172 sec
21.172 sec
1
2
3
4
5
6
Different Tests

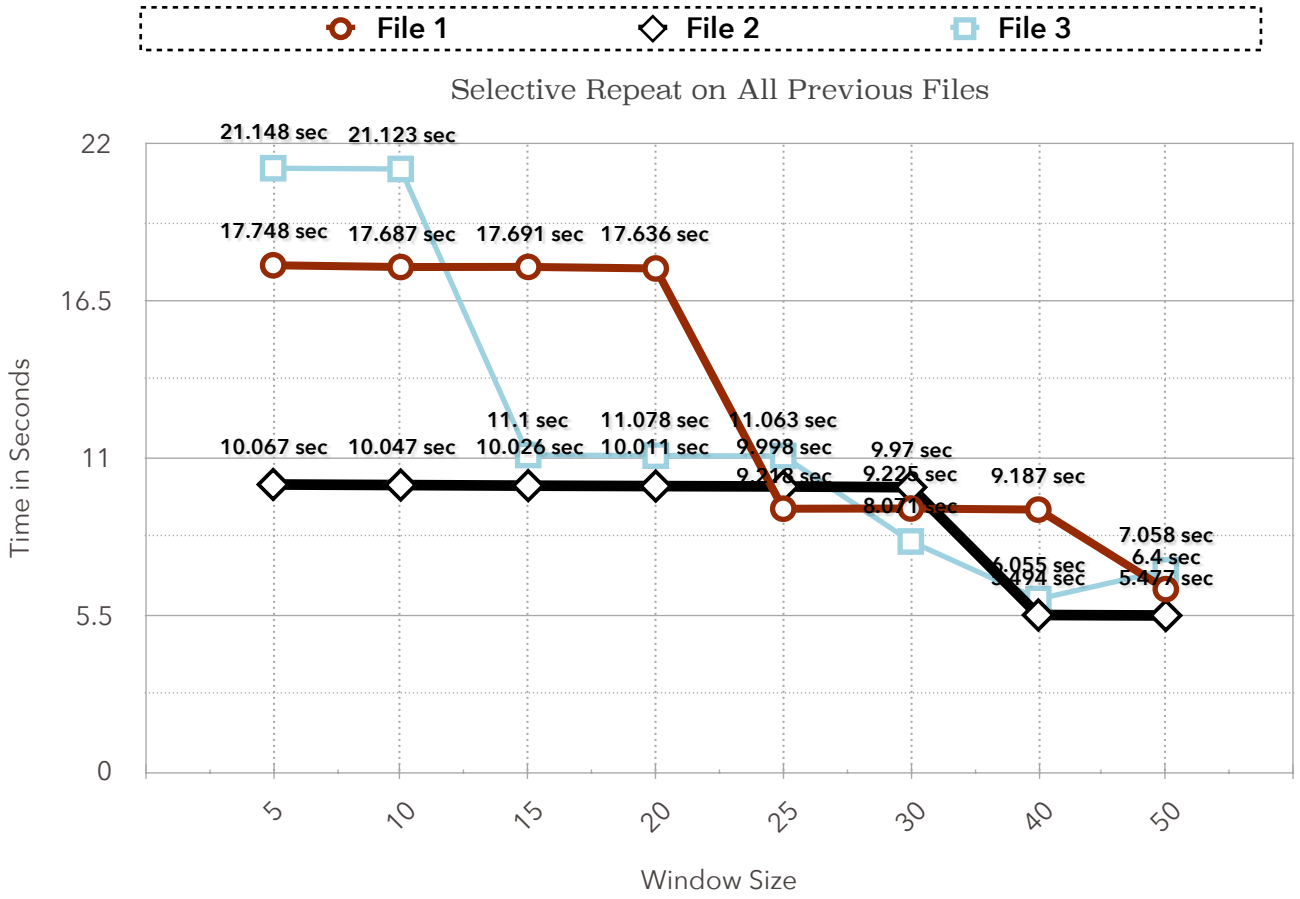

File 1
File 2
File 3
Stop and Wait on All Previous Files
25
20
15
10
5
Time in Seconds
21.192 sec
21.172 sec
21.176 sec
21.177 sec
21.172 sec
21.172 sec
17.767 sec
17.766 sec
17.765 sec
17.759 sec
17.764 sec
17.739 sec
10.088 sec
10.078 sec
10.084 sec
10.099 sec
10.082 sec
10.087 sec
1
2
3
4
5
6
Different Tests

✣ Selective Repeat:






Selective Repeat
File Size = 259080, Packet Size = 1000, Packets = 264
Wait Time = 1000 ms, Drop Rate= 8%
Time in Seconds
Window Size
22
16.5
11
5.5
0
5
10
15
20
25
30
40
50
21.148 sec
21.123 sec
11.1 sec
11.078 sec
11.063 sec
8.071 sec
6.055 sec
7.058 sec

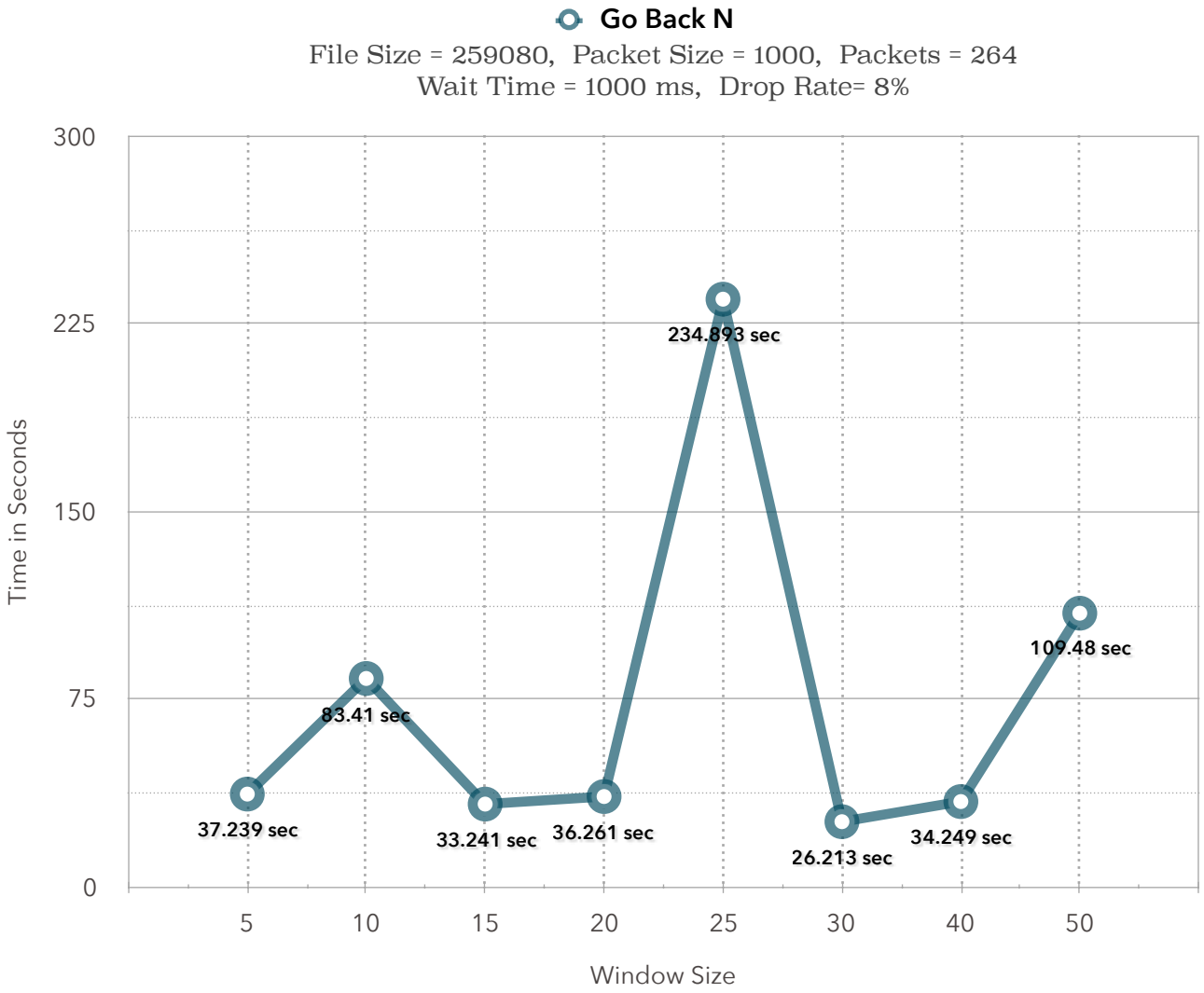

File 1
File 2
File 3
Selective Repeat on All Previous Files
Time in Seconds
Window Size
22
16.5
11
5.5
0
5
10
15
20
25
30
40
50
21.148 sec
21.123 sec
17.748 sec
17.687 sec
17.691 sec
17.636 sec
11.1 sec
11.078 sec
11.063 sec
10.067 sec
10.047 sec
10.026 sec
10.011 sec
9.998 sec
9.97 sec
9.225 sec
9.187 sec
7.058 sec
6.4 sec
5.477 sec

✤ Go Back N:

File 1: File Size = 496216, Packet Size = 1000, Packets = 505, Window Size = 20
Wait Time = 700 ms, Drop Rate= 5%.

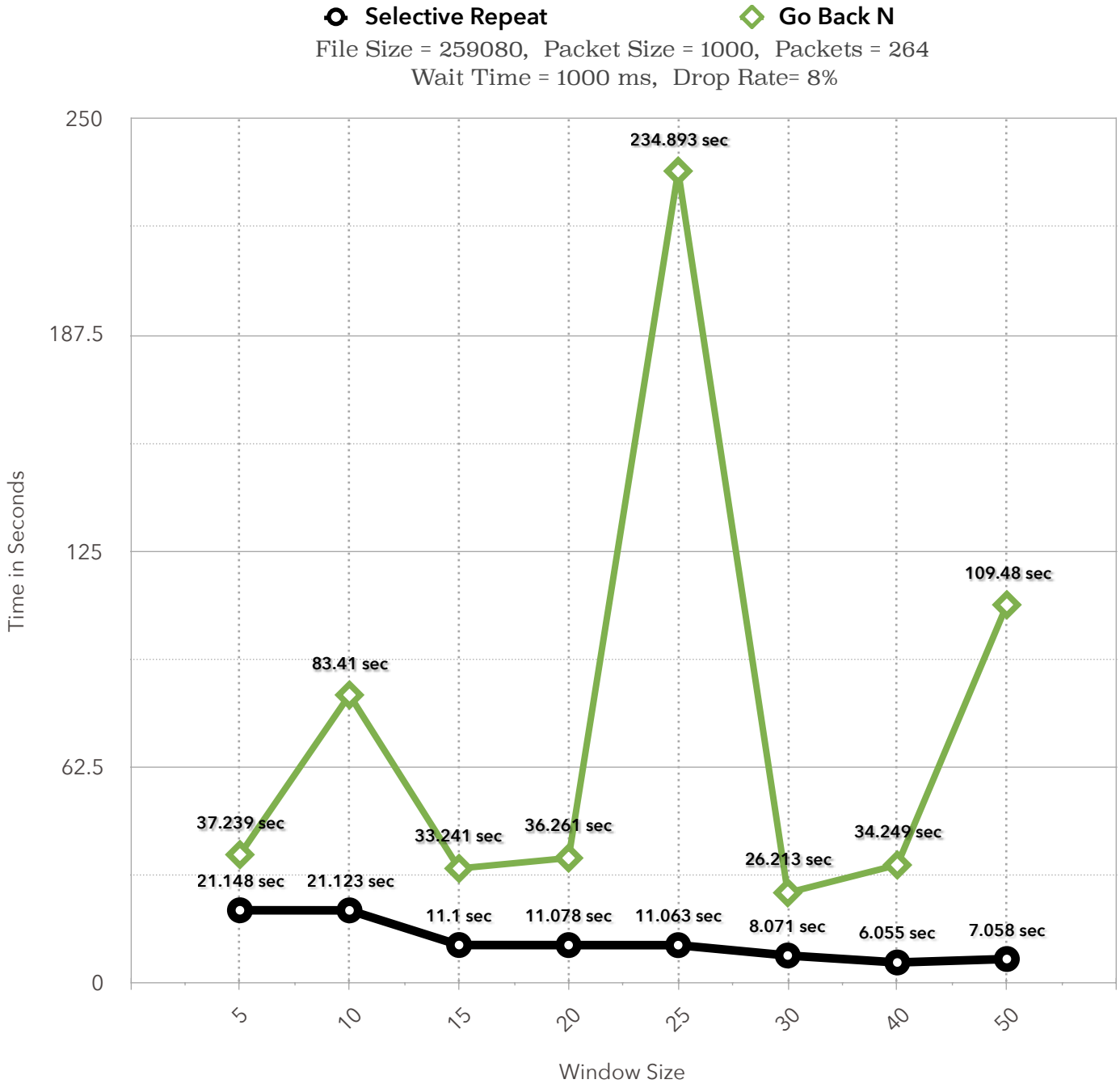
Time in seconded = 331.653 second (very slow).

File 2:

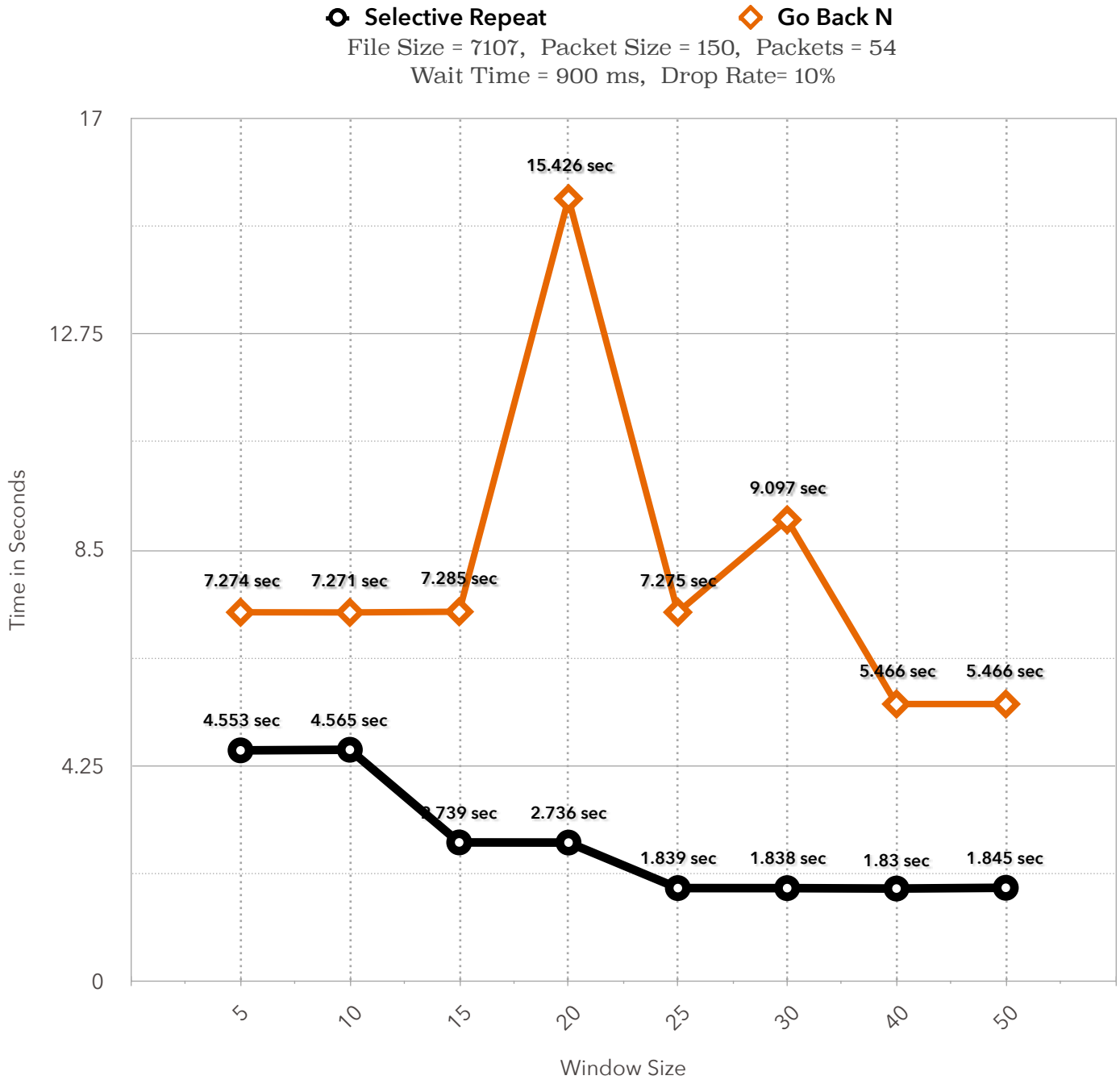


✤ Selective Repeat *VS* Go Back N:

✦ The same file with the same settings:



✦ The same file with the same settings:



Conclusion: *Selective Repeat* is the Fastest.