

## **CS352 - Internet Technology: Project 3**

**Submitted by Shelby Kurz**

**August 9th, 2019**

This pdf is comprised of three sections: 1. Stop-and-Wait, 2. Go-Back-N, and 3. Selective Repeat. Each section will describe the implementation of each mechanism and example output.

A .zip file accompanies this file, and is made up of three folders for each mechanism, each folder containing a Sender.py file and a Receiver.py file.

To run any of the mechanisms, open two command line programs, run the Sender.py first (input: `python sender.py`), then the Receiver.py file second (input: `python receiver.py`). The output is the acknowledgements received (sender side) and the messages received (receiver side)

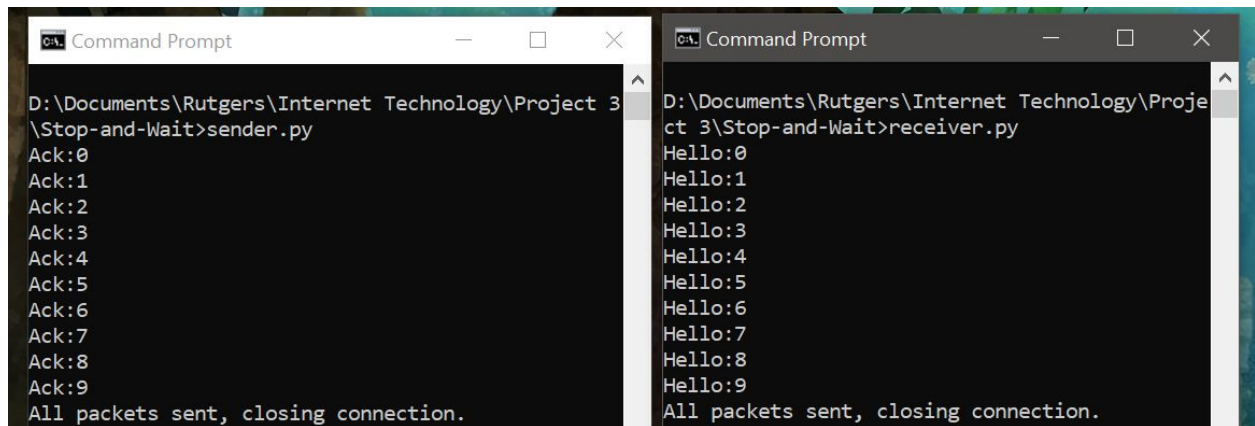
The code may be modified for each mechanism to allow for % packet loss rate (default: S&W(0%), GBN(30%), SR(30%)), number of packets, and window size. Details on modifying the code is included in each mechanism's description.

## 1. Stop-and Wait

- All algorithms will be using Python 3
- This algorithm allows the sender to send one packet at a time. The receiver will receive the sequence and send back a corresponding Ack. If the correct Ack was sent back, the sender will then proceed to sending the next packet. This repeats until all packets have been sent. At this point, the sender will send a disconnect message and both sides will close connection.
- Sender.py will send one packet at a time
  - $S\_Packet = Message + Sequence\ Number$
- Receiver will receive packet, print it, and send back a packet
  - $R\_Packet = Ack + Sequence\ Number$
- USER TOOLS
  - Sender.py, line 46: Uncomment this to see the packets that are being sent (example 3)
  - Line 25 (s) or 22 (r): Customizable Variables
    - Number of Packets to be sent
    - Message to be sent
    - Packet loss chance (s only)

- **Stop-and-Wait Examples**

Example 1:

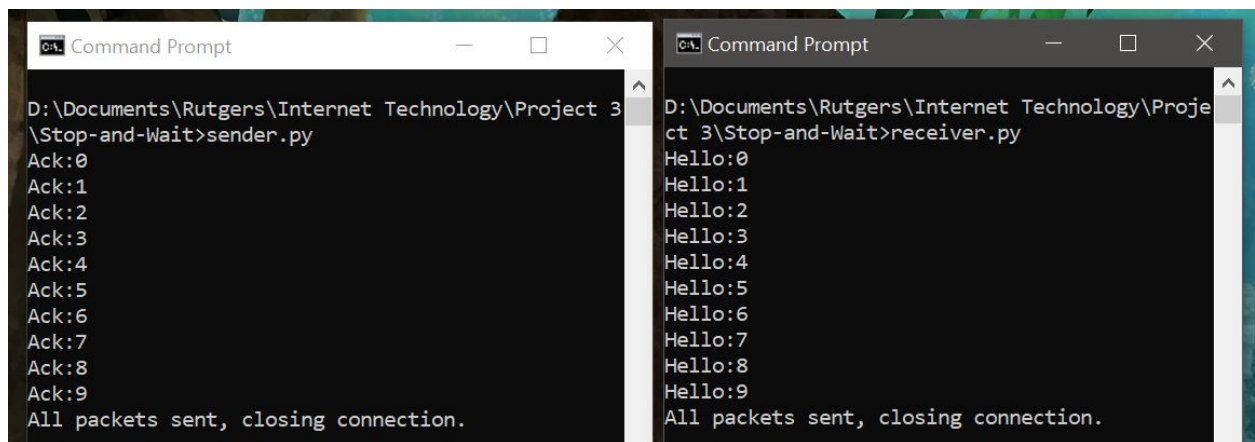


The image shows two side-by-side Command Prompt windows. The left window is titled 'Command Prompt' and shows the execution of 'sender.py' in the directory 'D:\Documents\Rutgers\Internet Technology\Project 3\Stop-and-Wait'. It outputs 'Ack:0' through 'Ack:9' and then 'All packets sent, closing connection.' The right window is also titled 'Command Prompt' and shows the execution of 'receiver.py' in the same directory. It outputs 'Hello:0' through 'Hello:9' and then 'All packets sent, closing connection.'

```
D:\Documents\Rutgers\Internet Technology\Project 3\Stop-and-Wait>sender.py
Ack:0
Ack:1
Ack:2
Ack:3
Ack:4
Ack:5
Ack:6
Ack:7
Ack:8
Ack:9
All packets sent, closing connection.

D:\Documents\Rutgers\Internet Technology\Project 3\Stop-and-Wait>receiver.py
Hello:0
Hello:1
Hello:2
Hello:3
Hello:4
Hello:5
Hello:6
Hello:7
Hello:8
Hello:9
All packets sent, closing connection.
```

Example 2:

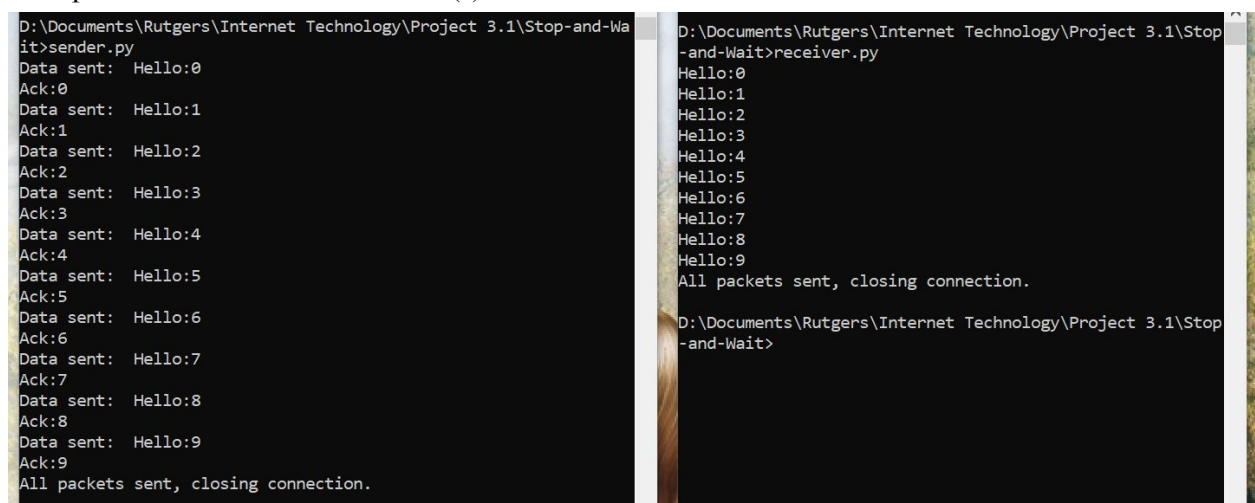


The image shows two side-by-side Command Prompt windows, identical to the ones in Example 1. The left window shows 'sender.py' outputting 'Ack:0' through 'Ack:9' and 'All packets sent, closing connection.' The right window shows 'receiver.py' outputting 'Hello:0' through 'Hello:9' and 'All packets sent, closing connection.'

```
D:\Documents\Rutgers\Internet Technology\Project 3\Stop-and-Wait>sender.py
Ack:0
Ack:1
Ack:2
Ack:3
Ack:4
Ack:5
Ack:6
Ack:7
Ack:8
Ack:9
All packets sent, closing connection.

D:\Documents\Rutgers\Internet Technology\Project 3\Stop-and-Wait>receiver.py
Hello:0
Hello:1
Hello:2
Hello:3
Hello:4
Hello:5
Hello:6
Hello:7
Hello:8
Hello:9
All packets sent, closing connection.
```

Example 3 with uncommented line 25(s):



The image shows two side-by-side Command Prompt windows. The left window shows 'sender.py' outputting 'Data sent: Hello:0' through 'Data sent: Hello:9' and 'Ack:0' through 'Ack:9', followed by 'All packets sent, closing connection.' The right window shows 'receiver.py' outputting 'Hello:0' through 'Hello:9' and 'All packets sent, closing connection.' Below this, the prompt 'D:\Documents\Rutgers\Internet Technology\Project 3.1\Stop-and-Wait>' is visible.

```
D:\Documents\Rutgers\Internet Technology\Project 3.1\Stop-and-Wait>sender.py
Data sent: Hello:0
Ack:0
Data sent: Hello:1
Ack:1
Data sent: Hello:2
Ack:2
Data sent: Hello:3
Ack:3
Data sent: Hello:4
Ack:4
Data sent: Hello:5
Ack:5
Data sent: Hello:6
Ack:6
Data sent: Hello:7
Ack:7
Data sent: Hello:8
Ack:8
Data sent: Hello:9
Ack:9
All packets sent, closing connection.

D:\Documents\Rutgers\Internet Technology\Project 3.1\Stop-and-Wait>receiver.py
Hello:0
Hello:1
Hello:2
Hello:3
Hello:4
Hello:5
Hello:6
Hello:7
Hello:8
Hello:9
All packets sent, closing connection.

D:\Documents\Rutgers\Internet Technology\Project 3.1\Stop-and-Wait>
```

## 2. Go-Back-N

- This algorithm allows a predetermined number of packets to be sent in a window, and the receiver will accept and send back an Ack for the packets in sequential order. If the receiver receives a packet that is not in the proper sequence, it will discard that packet and the remaining packets. The window will shift, starting with the first sequential packet that did not receive an Ack (last Ack number + 1), and restart the process. Once the Sender has received an Ack for all of the packets it wants to send, it will signal the Receiver to close and then Sender will close.
- Sender.py will send N packets at a time (N = window size, default 5)
  - To simulate the possibility of packets being dropped, there is a (packet\_loss\_chance)% that a given packet will be “lost” and will not reach the Receiver
  - Receiver can receive up to 5 packets per window
  - This percent chance can be set to 0 in the code for a 0% chance of packet loss (then it will just be up to five packets at a time, 5 Acks received, and repeat until all packets have been sent)
- Receiver will send back Acks in sequential order by keeping track of the last packet it received in sequence
  - If a packet is received out of sequence, Receiver will send back last sequential Ack
  - This repeats until the correct packet in the sequence is sent
  - Once the last packet (default 9th) has received Ack, both programs will close
- USER TOOLS
  - Sender.py, line 52: Uncomment this to see the packets that are being sent (example 3)
  - Line 29 (s) or 24 (r): Customizable Variables
    - Window Size
    - Number of Packets to be sent
    - Message to be sent
    - Packet loss chance (s only)

## • Go-Back-N Examples

<pre>D:\Documents\Rutgers\Internet Technology\Project 3 \Go-Back-N&gt;sender.py Ack: 0 Ack: 1 Ack: 1 Ack: 1 Ack: 1  Ack: 2 Ack: 3 Ack: 4 Ack: 5 Ack: 6  Ack: 7 Ack: 8 Ack: 9  All packets send, closing connection</pre>	<pre>D:\Documents\Rutgers\Internet Technology\Project 3\Go-Back-N&gt;receiver.py Hello:0 Hello:1  Hello:2 Hello:3 Hello:4 Hello:5 Hello:6  Hello:7 Hello:8 Hello:9  Now closing connection  D:\Documents\Rutgers\Internet Technology\Project 3\Go-Back-N&gt;</pre>
--	--

- **Example 1**
- Packets 0 and 1 arrive in correct sequence, but next packet was not packet 2
- Receiver sends back Ack: 1 to signal the last correct packet received
- Sender sends [2, 3, 4, 5, 6] in correct sequence, receives Ack back for all
  - Sender window had shifted to start with 2 since 0 and 1 had been received
- Sender sends [7, 8, 9] in correct sequence, receives Ack back for all
- Connection closes

<pre>D:\Documents\Rutgers\Internet Technology\Project 3 \Go-Back-N&gt;sender.py Ack: 0 Ack: 0 Ack: 0 Ack: 0 Ack: 0  Ack: 1 Ack: 2 Ack: 3 Ack: 4 Ack: 4  Ack: 5 Ack: 6 Ack: 7 Ack: 8 Ack: 9  All packets send, closing connection</pre>	<pre>D:\Documents\Rutgers\Internet Technology\Project 3\Go-Back-N&gt;receiver.py Hello:0  Hello:1 Hello:2 Hello:3 Hello:4  Hello:5 Hello:6 Hello:7 Hello:8 Hello:9  Now closing connection  D:\Documents\Rutgers\Internet Technology\Project 3\Go-Back-N&gt;</pre>
--	--

- **Example 2**
- Only Packet 0 arrived in correct sequence
- Receiver sends back Ack 0 to signal last correct packet received
- Sender sends [1, 2, 3, 4, x] but the 5th packet received “x” is not packet 5
- Receiver sends back Ack 1 - 4 and sends another Ack 4 to signal last correct packet received
- Sender sends [5, 6, 7, 8, 9]

- Receiver sends back Ack for all
- Connection closes

```

D:\Documents\Rutgers\Internet Technology\Project 3.1\Go-Back-N>sender.py
Data sent Hello:0 | Hello:1 | Hello:2 | Hello:3 |

Ack: 0
Ack: 1
Ack: 2
Ack: 3
Ack: 3
Data sent Hello:5 | Hello:6 | Hello:7 | Hello:8 |

Ack: 3
Ack: 3
Ack: 3
Ack: 3
Ack: 3
Data sent Hello:5 | Hello:6 | Hello:8 |

Ack: 3
Ack: 3
Ack: 3
Ack: 3
Ack: 3
Data sent Hello:4 | Hello:6 | Hello:8 |

Ack: 4
Ack: 4
Ack: 4
Ack: 4
Ack: 4
Data sent Hello:5 | Hello:6 | Hello:7 | Hello:8 | Hello:9 |

Ack: 5
Ack: 6
Ack: 7
Ack: 8
Ack: 9
All packets sent, closing connection.

D:\Documents\Rutgers\Internet Technology\Project 3.1\Go-Back-N>ack-N>receiver.py
Hello:0
Hello:1
Hello:2
Hello:3

Hello:4

Hello:5
Hello:6
Hello:7
Hello:8
Hello:9

All packets sent, closing connection.

D:\Documents\Rutgers\Internet Technology\Project 3.1\Go-Back-N>

```

- **Example 3**
- This has uncommented line 51 of Sender.py
- This allows the user to see the packets that are successfully being sent to Receiver, and the corresponding Acks that the receiver is sending back
- Note that the second and third group of packets being sent do not have Hello:4 because it got lost both times
- Chance of packet getting lost can be adjusted in Customizable Variables

### 3. Selective Repeat

- This algorithm allows for packets to be sent using a predetermined window size, like in the go-back-N algorithm, but allows for the buffering of packets received out of order. The receiver will send back an Ack for all received packets, and the Sender will send back any packets that did not get an Ack. If the Sender does not receive any Acks, it will repeat its attempt of sending the packets within the window. Once all packets in the window have received an Ack, the window will shift and the process will repeat until all packets have been sent.
- Sender.py will send a specific number of packets in groups of [window\_size] and wait for Receiver.py to send an Ack for each sent packet.
  - Possible packet loss is simulated during the message building segment of Sender.py.
  - If Sender.py does not receive all expected Acks, it will resend whichever packets that did not get an Ack.
  - Sender.py has a timeout feature: If it does not receive any Acks after a short period of time, it will attempt to resend whichever packets from the current set that still need Acks
- Receiver.py will accept packets in groups of [window\_size] and send back an Ack for each sequence number received that it was expecting
  - If not all expected sequence numbers were received, it will wait for Sender.py to send all missing packets before shifting window and expecting a new set
  - If repeat or unexpected packets are sent, they will be ignored and discarded
- USER TOOLS
  - Sender.py, line 69: Uncomment this to see the packets that are being sent (example 3)
  - Line 34 (s) or 25 (r): Customizable Variables
    - Window Size
    - Number of Packets to be sent
    - Message to be sent
    - Packet loss chance (s only)

## • Selective Repeat Examples

Project 3\Selective-Repeat>sender.py	Project 3\Selective-Repeat>receiver.py
Ack: 0	Hello: 0
Ack: 1	Hello: 1
Ack: 2	Hello: 2
Ack: 3	Hello: 3
Ack: 4	Hello: 4
Ack: 7	Hello: 7
Ack: 9	Hello: 9
Ack: 5	Hello: 5
Ack: 6	Hello: 6
Ack: 8	Hello: 8
Now closing connection	Now closing connection

- **Example 1**
- [0, 1, 2, 3, 4] were successfully sent and received Ack
- Window shifts
- [7, 9] were sent but [5, 6, 8] were lost
- [7, 9] receive Ack
- Sender resends [5, 6, 8] and they receive Ack

D:\Documents\Rutgers\Internet Technology\Project 3\Selective-Repeat>sender.py	D:\Documents\Rutgers\Internet Technology\Project 3\Selective-Repeat>receiver.py
Ack: 0	Hello: 0
Ack: 3	Hello: 3
Ack: 1	Hello: 1
Ack: 2	Hello: 2
Ack: 4	Hello: 4
Ack: 5	Hello: 5
Ack: 6	Hello: 6
Ack: 7	Hello: 7
Ack: 8	Hello: 8
Ack: 9	Hello: 9
Now closing connection	Now closing connection

- **Example 2**
- [0, 3] are sent and received Ack, [1, 2, 4] are lost
- [1, 2, 4] is resent and received Ack
- Window shifts
- [5, 6, 7, 8, 9] are sent and received Ack



```

D:\Documents\Rutgers\Internet Technology\Project 3.1\Selective-Repeat>sender.py

sending: Hello:0 | Hello:2 | Hello:3 | Hello:4 |
Ack: 0
Ack: 2
Ack: 3
Ack: 4

sending: Hello:1 |
Ack: 1

sending: Hello:5 | Hello:6 | Hello:8 | Hello:9 |
Ack: 5
Ack: 6
Ack: 8
Ack: 9

sending:
sending:

sending: Hello:7 |
Ack: 7
All packets have been sent, closing connection

```

```

D:\Documents\Rutgers\Internet Technology\Project 3.1\Selective-Repeat>receiver.py
Hello: 0
Hello: 2
Hello: 3
Hello: 4

Hello: 1

Hello: 5
Hello: 6
Hello: 8
Hello: 9

Hello: 7

All packets have been sent, closing connection

D:\Documents\Rutgers\Internet Technology\Project 3.1\Selective-Repeat>_

```

- **Example 3**
- This has uncommented line 69 of Sender.py
- This allows the user to see the packets that are successfully being sent to Receiver, then the follow up attempts to send the packets that did not get an Ack
- Trace
  - Sent [0, 2, 3, 4]
  - Ack [0, 2, 3, 4]
  - Sent [1]
  - Ack [1]
  - Sent [5, 6, 8, 9]
  - Ack [5, 6, 8, 9]
  - Sent [] (packet lost)
  - Sent [] (packet lost)
  - Sent [7]
  - Ack [7]