GO-BACK-N:

Go-Back-N ARQ

❏ packets transmitted continuously (when available) without waiting for ACK, up to N outstanding, unACK’ed packets

❏ a logically different sender timer association with each unACK’ed packet: extension of AB protocol Receiver:

❏ ACK packet if correctly received and in-order, pass to higher layer

❏ NACK or ignore (possibly discard) corrupt or out-of-order packet

Sender : if NACK received, or timeout, for packet n, begin resending from n all over again

❏ cumulative ACK: ACK of n implicitly acknowledges up through n

Go Back n is a connection oriented protocol in which the transmitter has a window of sequence numbers that may be transmitted without acknowledgment. The receiver will only accept the next sequence number it is expecting - other sequence nubmers are silently ignored.

The protocol simulation shows a time-sequence diagram with users A and B, protocol entities A and B that support them, and a communications medium that carries messages. Users request data transmissions with *DatReq(DATAn),* and receive data transmissions as *DatInd(DATAn)*. Data messages are simply numbered *DATA0*, *DATA1*, etc. without explicit content. The transmitting protocol sends the protocol message *DT(n)* that gives only the sequence number, not the data. Once sequence numbers reach a maximum number (like 7), they wrap back round to 0. An acknowledgement *AK(n)* means that the *DT* message numbered *n* is the next one expected (i.e. all messages up to but not including this number have been received). Since sequence numbers wrap round, an acknowledgement with sequence number 1 refers to messages 0, 1, 7, 6, etc. Note that if a *DT* message is received again due to re-transmission, it is acknowledged but discarded.

The protocol has a maximum number of messages that can be sent without acknowledgement. If this window becomes full, the protocol is blocked until an acknowledgement is received for the earliest outstanding message. At this point the transmitter is clear to send more messages.

SELECTIVE REPEAT:

Selective Repeat is a connection oriented protocol in which both transmitter and receiver have a window of sequence numbers.

The protocol simulation shows a time-sequence diagram with users A and B, protocol entities A and B that support them, and a communications medium that carries messages. Users request data transmissions with *DatReq(DATAn),* and receive data transmissions as *DatInd(DATAn)*. Data messages are simply numbered *DATA0*, *DATA1*, etc. without explicit content. The transmitting protocol sends the protocol message *DT(n)* that gives only the sequence number, not the data. Once sequence numbers reach a maximum number (like 7), they wrap back round to 0. An acknowledgement *AK(n)* means that the *DT* message numbered *n* is the next one expected (i.e. all messages up to but not including this number have been received). Since sequence numbers wrap round, an acknowledgement with sequence number 1 refers to messages 0, 1, 7, 6, etc. Note that if a *DT* message is received again due to re-transmission, it is acknowledged but discarded.

The protocol has a maximum number of messages that can be sent without acknowledgement. If this window becomes full, the protocol is blocked until an acknowledgement is received for the earliest outstanding message. At this point the transmitter is clear to send more messages.

The receiver delivers the protocol messages *DT(n)* to the user in order. Any received out of order, but within the receiver's window are buffered.

Command Prompt Commands:

SENDER: java Sender asd.txt 9876 1000

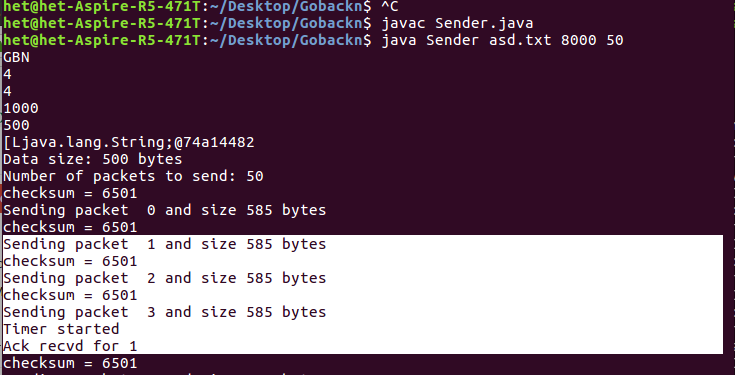
RECEIVER: java Receiver asd.txt 9876

GO-BACK-N:

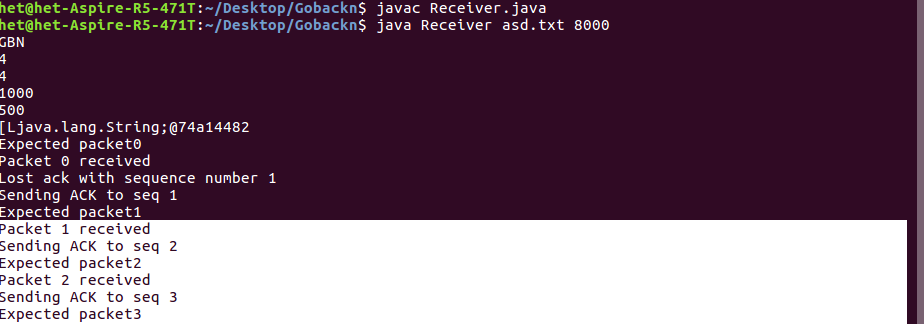
**Ideal Condition:**

**ScreenShots:**

**Sender (ideal)**

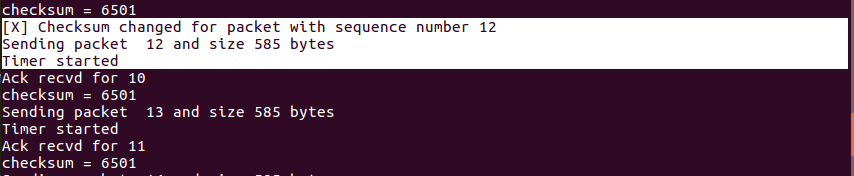
****

**Receiver(ideal)**

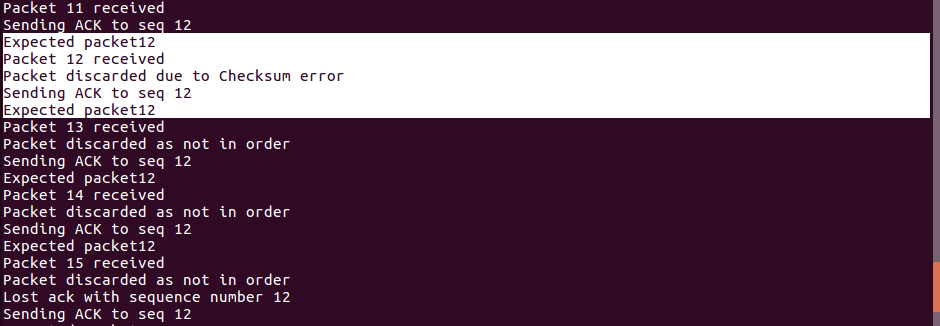
****

Checksum error

Sender

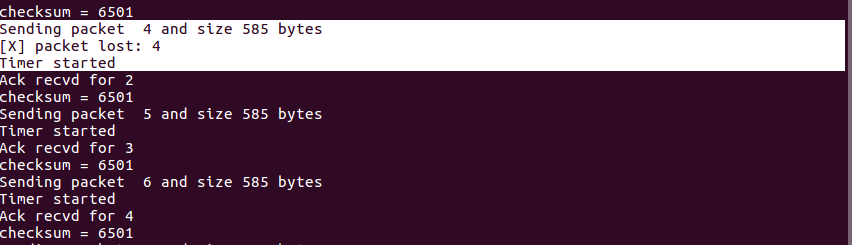


Check sum error receiver



Packet loss

Sender



;

receiver



Ack loss:

Sender



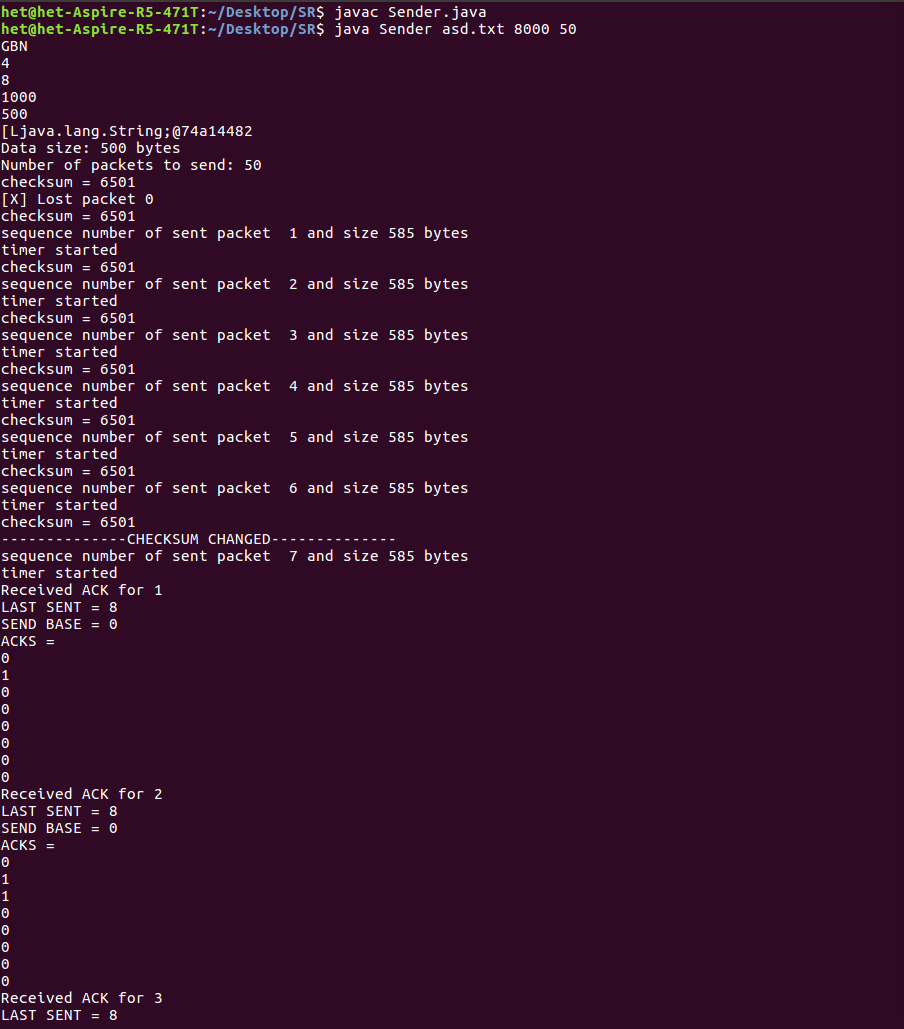
Receiver:



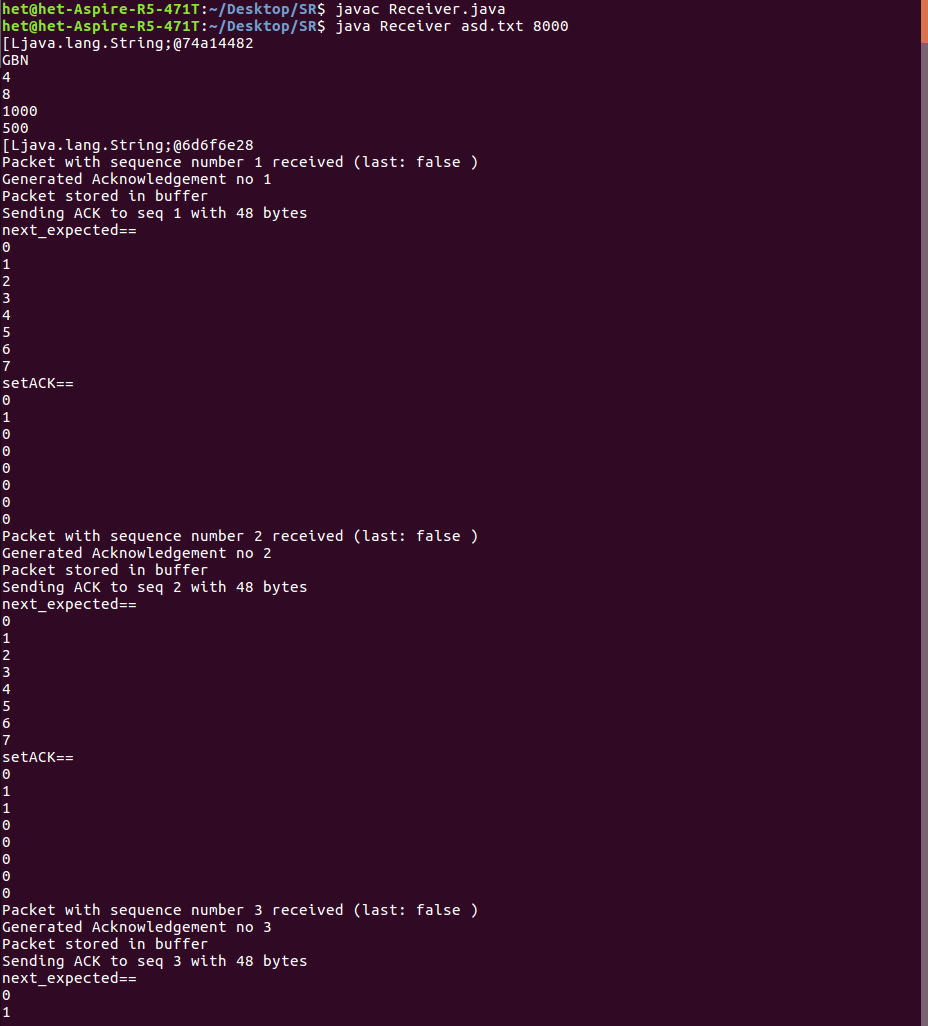
SELECTIVE REPEAT

**Ideal Condition:**

**sender**

****

**Receiver**

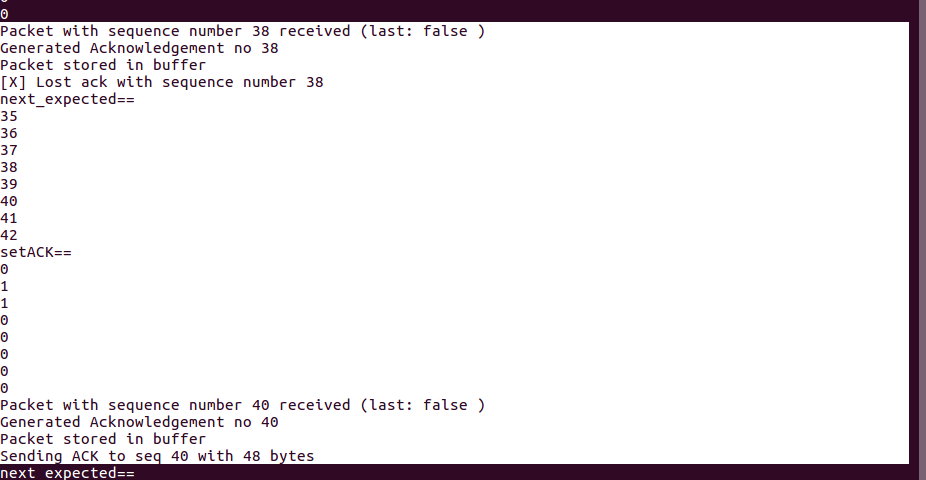
****

**Packet loss**

**Sender**

****

**Receiver**

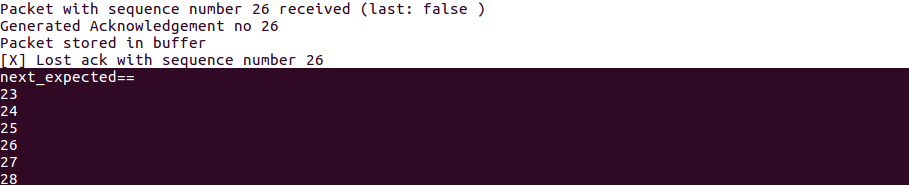
****

**Ack loss**

**Sender**

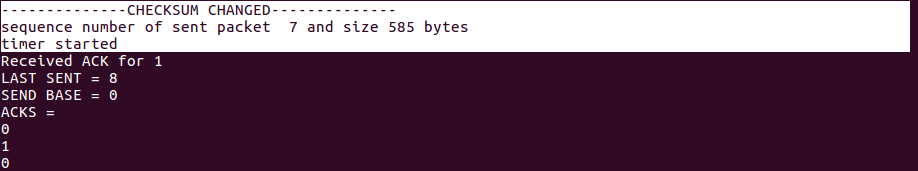
****

**Receiver**

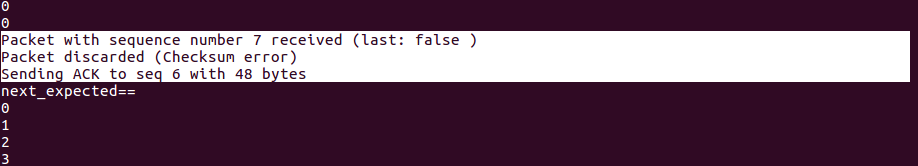
****

**Checksum**

**Sender**

****

**Receiver:**

****