NAME:- APURVA KUMAR GUPTA
USN :- 1CR18CS028
SEC :- 5-A

Page No.
Date

1) A TCP connection is always point to point communication, i.e. between a single sender and a single receiver so called 'Multicasting' - the transfer of data from one sender to many receivers in a single send operation is not possible with TCP. With TCP, two Ports are company and three are a crowd.

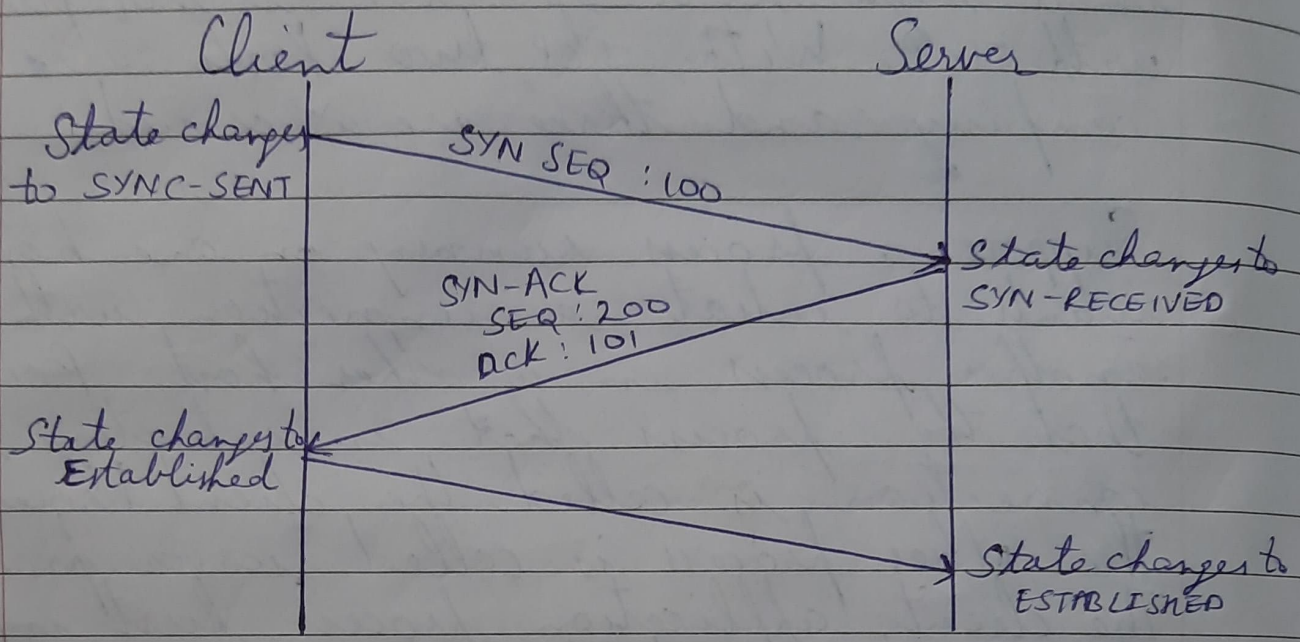
Suppose a process running in one host wants to initiate a connection with another process in another host. Recall that the process that is initiating the connection is called the client Process, while the other process is called server process. The client application process first informs the client transport layer that it wants to establish a connection to a process in the server.

Eg:- clientSocket.connect((ServerName, ServerPort))

TCP in the client then proceeds to establish a TCP connection with TCP in the server. Client first sends a special TCP segment, and finally the client responds again with a third special segment.

The first two segments carry no payload, that is no-application layer data. The third of these segment carry a payload. Because

three segments are sent between the two hosts, this connection-establishment procedure is often referred to as a "three-way handshake". Once a TCP connection is established, the two application processes can send data to each other.



2) TCP's use of sequence numbers reflects this view in that sequence numbers are over the stream of transmitted bytes and not over the series of transmitted segments. The sequence number for a segment is therefore the byte-stream number of the first byte in the segment.

Go-Back-N (GBN)

* In a Go-Back-N (GBN) protocol, the sender is allowed to transmit multiple packets without waiting for an acknowledgment, but is constrained to have no more than

some maximum allowable number, N, of unacknowledged packets in the pipeline.

* The base is the sequence number of the oldest unacknowledged packet and nextseqnum is the smallest unused sequence number. Sequence numbers in the interval [0, base-1] correspond to packets that have already been transmitted and acknowledged.
* The interval [base, nextseqnum-1] corresponds to packets that have been sent but not yet acknowledged.
* Sequence numbers in the interval [nextseqnum, base+N-1] can be used for packets that can be sent immediately, should data arrive from the upper layer.
* Sequence number greater than or equal to base+N cannot be used until an acknowledged packet currently in the pipeline has been acknowledged.
* The range of permissible sequence numbers for transmitted but not yet acknowledged packets can be viewed as a window of size N over the range of sequence numbers. N is often reffered to as the window size and GBN protocol itself as a sliding window protocol.

3) → The TCP segment consists of header fields and a data field.
→ The data field contains a chunk of application data.
→ The minimum length of TCP header is

20 bytes.

→ The header includes source and destination port numbers which are used for multiplexing/demultiplexing data from/to upper layer applications.

→ The header includes a checksum field for error detection.

→ A TCP segment header also contains the the following fields

i) The 32 bit sequence number field and the 32 bit acknowledgment number field are used by the TCP sender and receiver in implementing a reliable transfer service.

ii) The 16 bit receiver window is used for flow control. It is used to indicate the number of bytes that a receiver is willing to accept.

iii) The 4 bit header length field specifies the length of the TCP header in 32 bit words. The TCP header can be of variable length due to the TCP option field.

iv) The optional and variable length field is used when a sender and receiver negotiates the Maximum Segment Size (MSS) or as a window scaling factor for use in high speed networks.

v) The flag field contains 6 bits

vi) The ACK bit is used to indicate that the value carried in the acknowledgment for a segment that has been successfully received.

vii) The RST, SYN, FIN bits are used for

<table>
<tr><td colspan="4">Source port #</td><td colspan="4">Dest port #</td></tr>
<tr><td colspan="8">Sequence number</td></tr>
<tr><td colspan="8">Acknowledgment number</td></tr>
<tr><td>Header length</td><td>Unused</td><td>URG</td><td>ACK</td><td>PSH</td><td>RST</td><td>SYN</td><td>FIN</td><td>Receive window</td></tr>
<tr><td colspan="6">Internet checksum</td><td colspan="2">Urgent data pointer</td></tr>
<tr><td colspan="8">Options</td></tr>
<tr><td colspan="8">Data</td></tr>
</table>

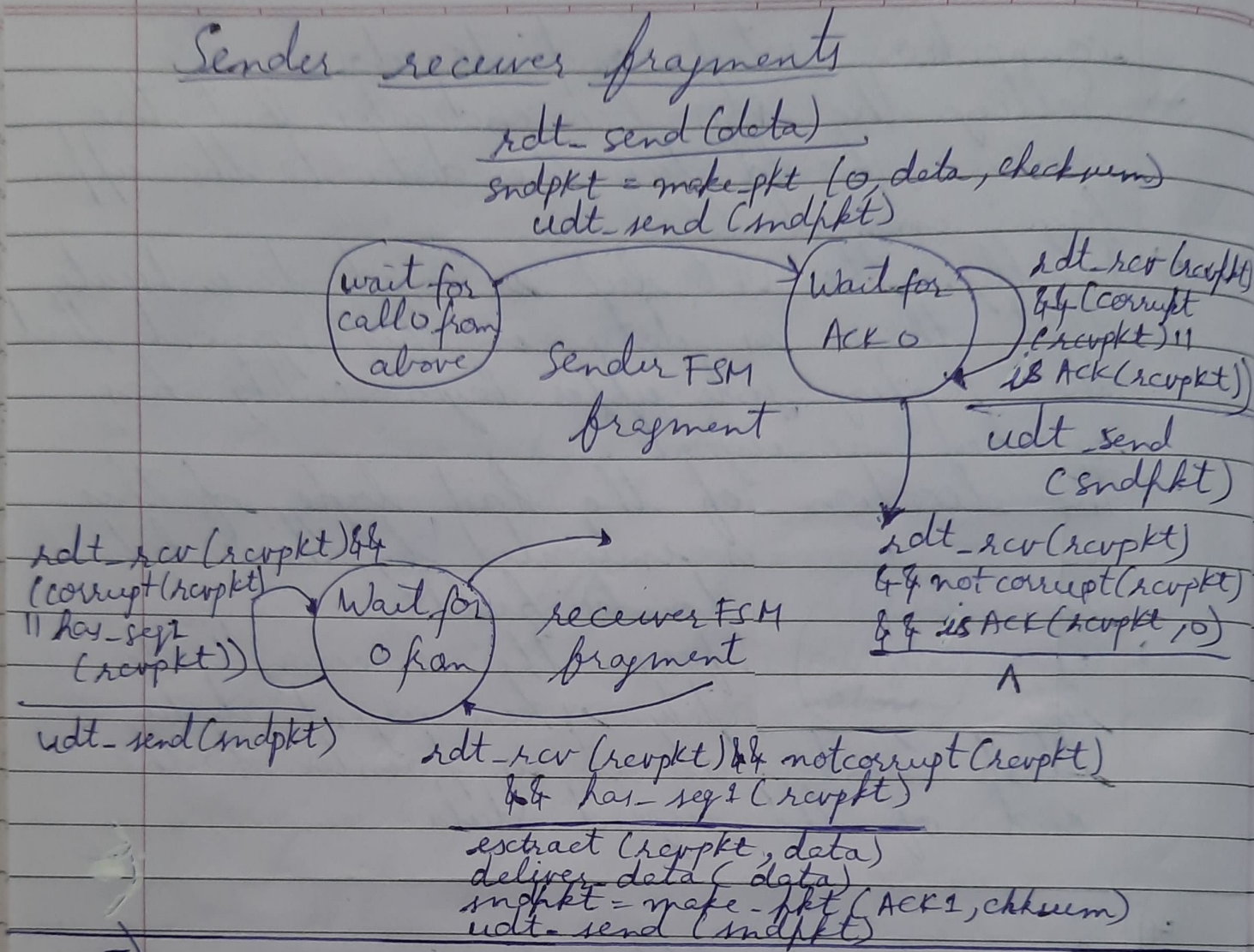
connection setup and teardown.
→ Setting the PSH bit indicates that the receiver should pass the data to the upper layer immediately.
→ Finally, the URG bit is used to indicate that there is data in this segment that the sending-side upper layer entity has marked as urgent.
→ The location of the last byte of this urgent data is indicated by the 16 bit urgent data pointer field.
→ TCP must inform the receiving-side upper layer entity when urgent data exists and pass it a pointer to the end of the urgent data

---

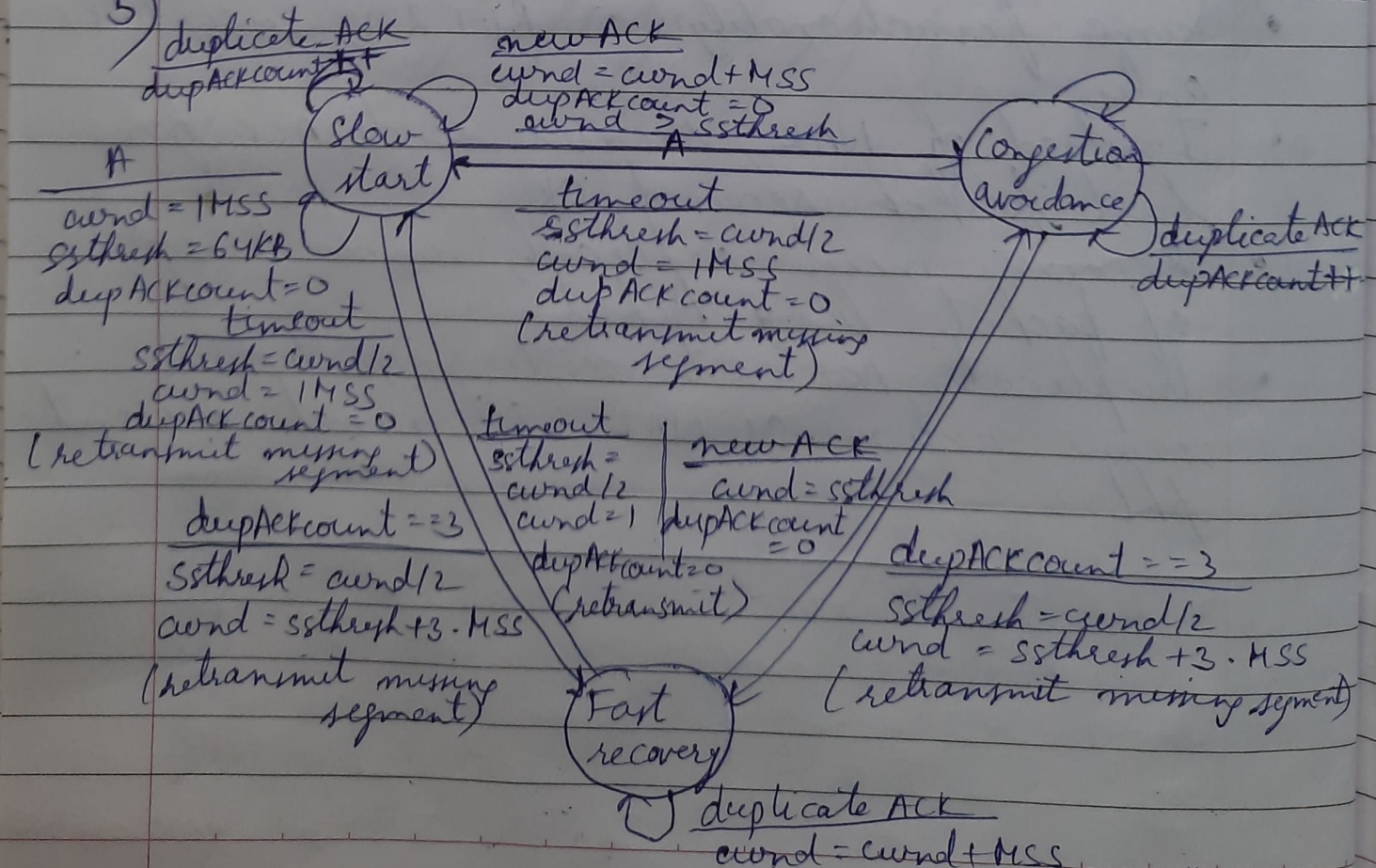
4)
- Same functionality as RDT 2.1, using ACK's only.
- Instead of NAK, receiver sends ACK for last packet received OK
- Receiver must explicitly include seq# of packet being Acked.
- Duplicate ACK at sender results in same action as NAK: retransmit current pkt.

Page No.
Date

Sender receiver fragments



5)



Page No.
Date / /

Slow start:

* When a TCP connection begins, the value of cwnd is typically initialized to a small value of 1 MSS, resulting in an initial sending rate of roughly MSS/RTT. The available bandwidth to the TCP sender may be much larger than MSS/RTT, the TCP sender would like to find the amount of available bandwidth quickly. Thus, in the slow-start state, the value of cwnd begins at 1 MSS and increases by 1 MSS every time a transmitted segment is first acknowledged.
* TCP sends the first segment into the network and waits for an acknowledgment. When this acknowledgment arrives, the TCP sender increases the congestion window by one MSS for each of the acknowledgment segments, giving a congestion window of 4 MSS, and so on. This process results in a doubling of the sending rate every RTT. Thus the TCP send rate starts slow but grows exponentially during the slow start phase.

Congestion avoidance:

* On entry to the congestion-avoidance state, the value of cwnd is approximately half its value when congestion was last encountered - congestion could be just around

the corner. Thus rather than doubling the value of cwnd every RTT, TCP adopts a more conservative approach and increases the value of cwnd by just a single MSS every RTT. A common approach is for the TCP sender to increase cwnd by MSS bytes, whenever a new acknowledgment arrives.

Fast recovery:

* The value of cwnd is increased by 1 MSS for every duplicate ACK received for the missing segment that caused TCP to enter the fast recovery state. Eventually when an ACK arrives for the missing segment, TCP enters the congestion-avoidance state after deflating cwnd. If a timeout event occurs, fast recovery transitions to the slow-start state after performing the same actions as in slow-start and congestion avoidance. The value of cwnd is set to 1 MSS, and the value of ssthresh is set to half the value of cwnd when loss of event occured.

---

6) TCP socket is identified by a four-tuple (Source IP address, source port number, destination IP address, destination port number). Thus, when a TCP segment arrives from the network to a host, the host uses all four values to direct (demultiplex) the

segment to appropriate socket. In particular and in contrast with UDP, two arriving TCP segments with different source IP addresses or source port number will be directed to two different sockets

* The TCP server application has a "welcoming socket" that waits for connection establishment requests from TCP clients on port number 12000 and a special connection-establishment bit set in the TCP header. The segment also includes a source port number that was chosen by client.
* When the host operating system of the computer running the server process receives the incoming connection-request segment with destination port 12000, it locates the server process that is waiting to accept a connection on port number 12000. The server then creates a new socket

connectionSocket, addr = serverSocket.accept()

* Also, the transport layer at the server notes the following four values in the connection-request segment:
  (i) the source port number in the segment
  (ii) the IP address of the source host
  (iii) the destination port number in the segment
  (iv) its own IP address.
* The newly created connection socket is identified by these four values.