Assignment 02 – Submission 01

Group Members

- Nauman Haroon - BESE-3(B) - 01917
- Umair Ahmad - BESE-3(B) – 00321

Description

The protocol used is ***STOP AND WAIT*** protocol.

Handling BadNet0: (Case: No Error)

- Sender reads data from the file in chunks and creates a list of packets
- While loop for checking, if list has any packet then send it to the receiver
- Sender sends one packet and waits for its ACK, on receiving ACK for that packet sender removes that packet from the packet list and then sends the next packet in the list and so on
- Receiver has a loop for receiving the packets until it receives the last packet
- Receiver receives one packet sends its ACK and then wait for the next packet
- After receiving all the packets receiver writes it in the file and save that file in current working directory

Handling BadNet1: (Case: if a packet drops)

- For Packet Drop we have used setimeout(time in secs) function. We have associated a timeout with the senders socket. It waits for the ACK of the sent packet for a certain time.
- If ACK is not received within that time limit then the sender retransmits that packet. (packet drop means no ACK, No ACK means timeout, timeout means Resend)
- The ACK dropped is also handled with the same timeout (ACK dropped mean No ACK, No ACK means timeout, timeout means Resend)
- We have used the time library for this purpose
- The function that we have used is *Socket.settimeout(n)* ---- It associates a timeout of n secs with the socket (means that the socket will wait for n sec for a message to get received)

Handling BadNet2: (Case: if packet gets corrupted)

- For this case we have appended the checksum with the packet on sending. Now sender calculates the checksum, append it in the packet and then sends it to the receiver.
- Receiver on receiving calculates the packet's checksum (without the checksum that was appended) and then matches the calculated checksum with the received checksum if both are equal only then it sends an ACK. (sending the ACK will also includes the Checksum appending procedure)

- If packet gets corrupted the Sender resends that packet (corrupt packet means checksum will not get match at receiver which means that no ACK will be sent by receiver which means a timeout has occur and thus resending of the packet by the sender will take place)
- If ACK gets corrupted the packet is resent by the sender (Corrupted ACK will raise the condition of resending)
- we have used the hashlib library for calculating checksum

Handling BadNet3: (Case: Duplicate Packets)

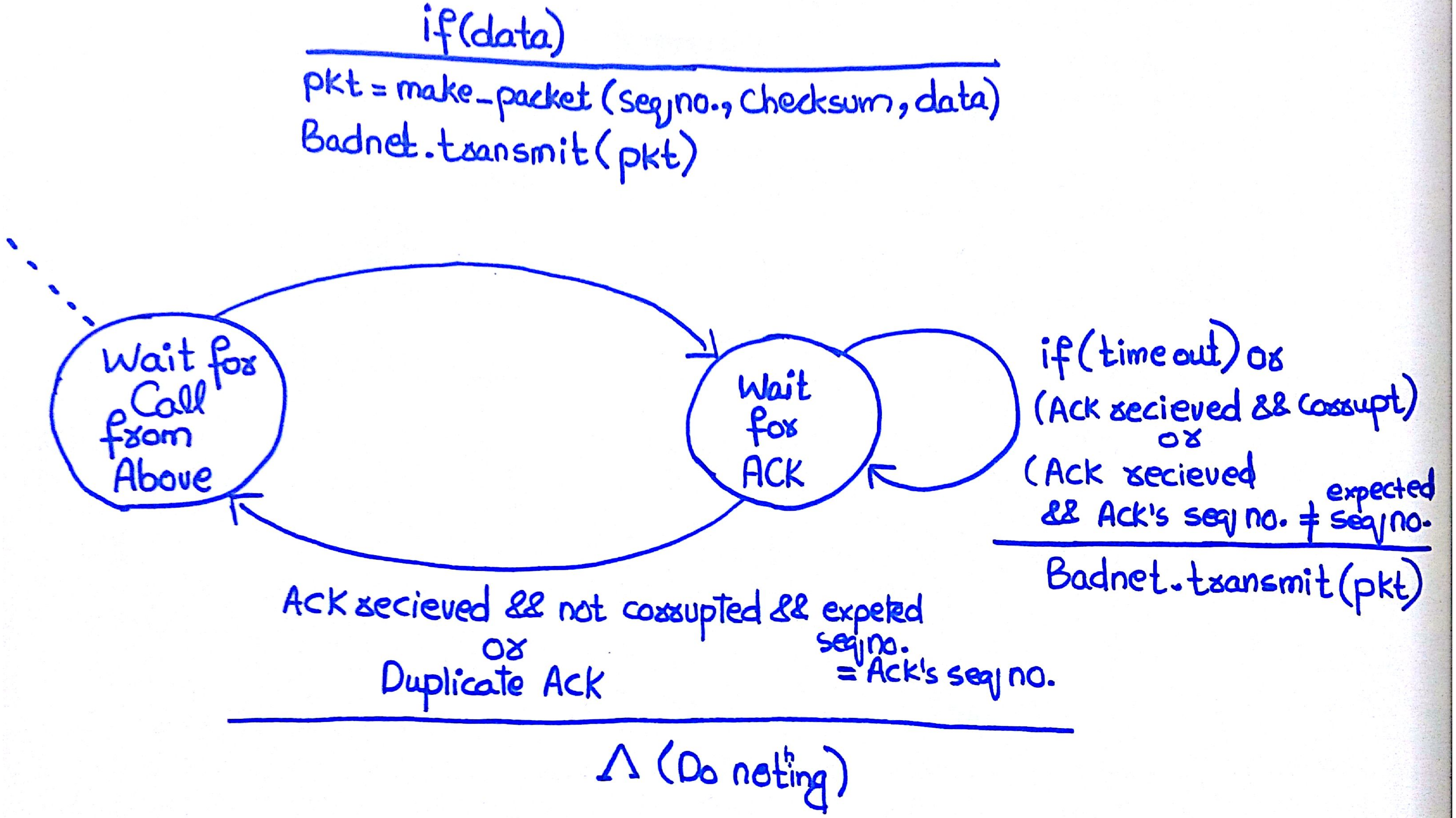
- we have maintained a list in which we are storing the sequence number of the packets that we are receiving, So whenever a packets is received we check if it is in that list or not
- if the received packet is not in that list then we store that packet and send an ACK for it.
- if the received packet is in the list then we do not store that packet but we do send an ACK for it.

Handling BadNet4: (Case:Reordering)

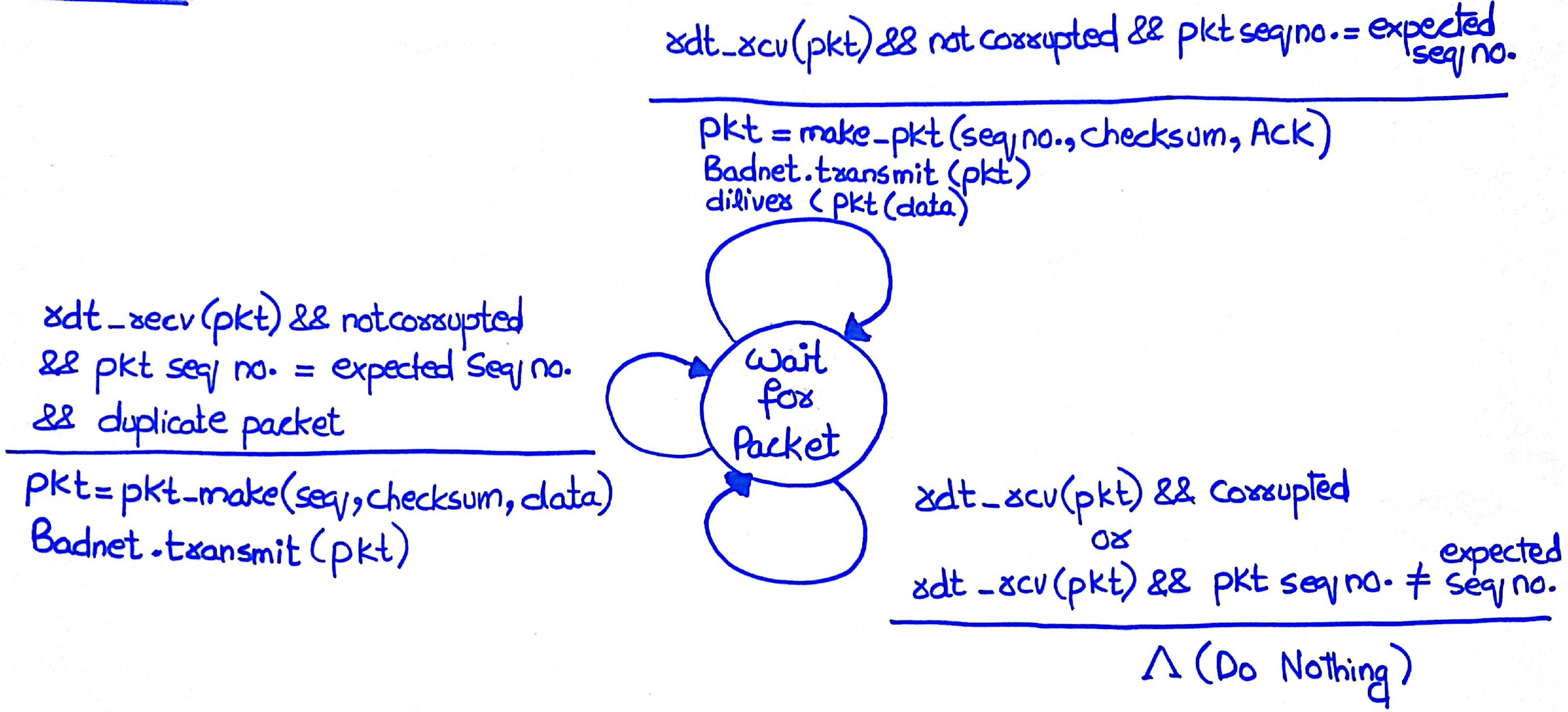
- we have associated sequence number with each packet.
- Whenever we receive a packet we check its sequence number.
- If received packet's sequence number is equal to the expected sequence number only then we store that packet
- if received packet's sequence number is not equal to the expected sequence number then we donot store it and send and ACK for the last received packet.

Functions used are:

- *checksum = hashlib.md5()* ---- Creates an Object for accessing the functions for calculating checksum (checksum is only a variable name, may vary)
- *checksum.update(serialized packet)* ---- Function that calculates the checksum of the packet that has passed to it
- *checksum.digest()* ---- Returns the value of the checksum that was calculated in the previous step


if(data)
pkt = make_packet (seq no., Checksum, data)
Badnet.transmit(pkt)
Wait for Call from Above
Wait for ACK
if(time out) or
(Ack recieved && corrupt)
or
(Ack recieved
&& Ack's seq no. ≠ expected seq no.
Badnet.transmit(pkt)
Ack recieved && not corrupted && expeted seq no. = Ack's seq no.
or
Duplicate Ack
Λ (Do nothing)

SERVER:



rdt_rcv(pkt) && not corrupted && pkt seq no. = expected seq no.

pkt = make_pkt (seq no., checksum, ACK)
Badnet.transmit (pkt)
diliver (pkt (data))

rdt_recv (pkt) && notcorrupted && pkt seq no. = expected Seq no. && duplicate packet

pkt=pkt-make(seq, checksum, data)
Badnet.transmit (pkt)

rdt_rcv(pkt) && Corrupted
or
rdt_rcv(pkt) && pkt seq no. ≠ expected seq no.

Λ (Do Nothing)