COMPUTER COMMUNICATION
PROJECT 01 REPORT

STUDENT NAME : HÜSEYİN ERDOĞAN
STUDENT NO : 040100054

STUDENT NAME : RESUL YAYKIRAN
STUDENT NO : 040100013

CRN : 12337

BLG433E PROJECT 01 REPORT

1 Stop-and-Wait Protocol Data Structers

-
  ```
  class my_frame{
          int frame_id; //used for storing frame’s id
          bool has_transmission_error; //used for controlling is there a error in frame
          bool has_ack_error; //used for controlling is there a error in acknowledge

          my_frame(); // constructor for this class and equalize both has_transmission_error and has_ack_error  to “false”
          void set_transmission_error(bool t); // used for setting a value to has_transmission_error
          void set_ack_error(bool t); // used for setting a value to has_ack_error

  }
  ```

-
  ```
  class my_event{
          char station; // used to store station type 'A' or 'B'
          my_frame t_frame; // used to store which frame is used in this event
          int event_time; // used to store time when this event is happening
          int is_transmitted; // is frame transmitted succesfully
          int is_received; // is frame received succesfully

          my_event(); // used for equalize is_transmitted and is_received to zero
          void set_is_transmitted(int t); // used for setting a value to is_transmitted
          void set_is_received(int t); // used for setting a value to is_received;
          void set_event_time(int t); // used for setting a value to event_time
          void set_station(char t); // used for setting a value to station
  }
  ```

-
  ```
  class stop_and_wait{
          vector<my_frame> frame_array; //used to store frames
          vector<my_event> my_event_list; // used to store every event both transmitting and receiving
          int starting_time; //starting time of our transmission
          int ending_time; //time that shows when our transmission is ended
          int number_of_frame; //used for storing number of frame

          stop_and_wait(int t_size,int t_starting_time); //this function is a constructor and used for setting starting_time and number_of_frame
          my_event last_event(); // used for returning last event in our event list
          void station_A(); // used for transmitting
          void station_B(); //used for receiving
          bool is_finished(); //used for controlling our transmission is ended or not
          void print_event_list(); //used for printing event list
          void rand_function(int error_rate); //used for setting random errors to our frame array
  }
  ```

2 Stop-and-Wait Protocol Implementation

- Psuedo Code of station_A

```
my_event t_event;
if my_event_list is empty{
    Set time to t_event
    Set transmitted frame to t_event
    if frame has an error{
        Set is_transmitted of t_event to ‘1’
    }
    else{
        Set is_transmitted of t_event to ‘0’
    }

}
else{
    if last frame transmitted successfully{
        take next frame;
        set time to event
        if frame has an error{
            Set is_transmitted of t_event to ‘1’
        }
        else{
            Set is_transmitted of t_event to ‘0’
        }
    }
    else{
        take last frame
        set a timeout to event
        Set is_transmitted of t_event to ‘1’
    }
}
push t_event to my_event_list vector
```

- Psuedo Code of station_B

```
my_event t_event;
take frame of last event
set a time to event
if last frame received AND last frame has not a ack error{
    Set is_received of t_event to ‘1’
}
else{
    Set is_received of t_event to ‘0’
}
push t_event to my_event_list vector
```

3 Selective Repeat Implementation

- Psuedo Code of Station_A

```
data from above
if next available sequence in window{
        send frame
}

timeout(n)
resend frame n
restart timer

ACK(n)
mark frame n as received
if n smallestframe that not used by ACK(n){
        advance window base to next not ACK(n) sequence
}
```

- Psuedo Code of Station_B

```
frame n in [rcvbase, rcvbase+N-1]
send ACK(n)
out-of-order :  buffer
in-order : deliver, advance window to next not-yet received frame

frame n in [rcvbase-N,rcvbase-1]
ACK(n)

otherwise
  ignore
```

4 Explanation of Frame Series in Test Set

4.1 Test 1 – Stop and Wait

- Station A sends Frame 1
- Station B receives Frame 1 and send ack for Frame 2
- Station A receives ack for Frame 2 and sends Frame 2
- Station B receives Frame 2 and send ack for Frame 3
- Station A receives ack for Frame 3 but can not send Frame 3 because of error
- Timeout expires Station A sends Frame 3 again
- Station B receives Frame 3 and send ack for Frame 4
- Station A receives ack for Frame 4 and sends Frame 4
- Station B receives Frame 4 and send ack for Frame 5
- Station A receives ack for Frame 5 and sends Frame 5
- Station B receives Frame 5 and send ack for Frame 6

- Station A receives ack for Frame 6 and sends Frame 6
- Station B receives Frame 6 and send ack for Frame 7
- Station A receives ack for Frame 7 and sends Frame 7
- Station B receives Frame 7 and send ack for Frame 8
- Station A receives ack for Frame 8 and sends Frame 8
- Station B receives Frame 8 and send ack for Frame 9
- Station A receives ack for Frame 9 and sends Frame 9
- Station B receives Frame 9 and send ack for Frame 10
- Station A receives ack for Frame 10 and sends Frame 10
- Station B receives Frame 10 and send ack
- Transmission ends.

4.2 Test 2 – Selective Repeat

- Station A sends Frame 1
- Station B receives Frame 1 and send ack for Frame 2
- Station A receives ack for Frame 2 and sends Frame 2
- Station B receives Frame 2 and send ack for Frame 3
- Station A receives ack for Frame 3 but can not send Frame 3
- Frame 3 lost and can not be received by station B.
- Whenever an out-of-sequence frame is observed at the station B, an NAK frame is sent by station B with number of expected frame, frame 3 (NAK3)
- When station A receives NAK frame, it transmits Frame 3 again.
- When waiting for Frame 3, station B receives Frames 4,5,6,7 and buffers them
- After station A transmits Frame 3, station transmits Frame 8,9,10
- Station B receives Frame 8,9,10 and sends ack to Station A.
- Transmission ends.

4.3 Test 3 – Selective Repeat

- Station A sends Frame 1
- Station B receives Frame 1 and send ack for Frame 2
- Station A receives ack for Frame 2 and sends Frame 2
- Station B receives Frame 2 and send ack for Frame 3
- Station A receives ack for Frame 3 but can not send Frame 3
- Frame 3 lost and can not be received by station B.
- Whenever an out-of-sequence frame is observed at the station B, an NAK frame is sent by station B with number of expected frame, frame 3 (NAK3)
- Station A receives ack error before so it retransmits all Frames that are transmitted after Frame 3..
- Station B receives Frames 3,4,5,6,7,8,9,10
- Station B sends ack for Frames 3,4,5,6,7,8,9,10 to station A.
- Transmission ends.

5 Size of the sliding window in Technique 2

In the test 2 set, the sender was sending 4 frames after error occured when sending frame Therefore the windows size is 4 for that table. Source ; http://www.cse.iitk.ac.in/users/dheeraj/cs425/lec09.html

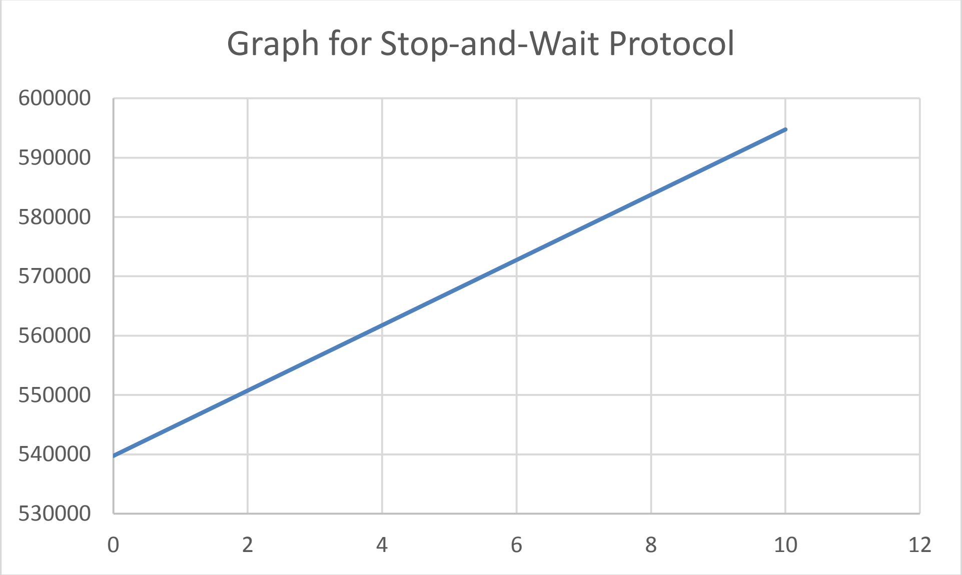
6 Size of buffer in Technique 2

Size of buffer have to be same with size of sliding window int Technique 2.

7 Maximum window size for Technique 2

Sequence number is 3 bit so the max window size=2^(n-1)=2^2=4

8 Graph for presenting relationship between error rate and simulation time


Graph for Stop-and-Wait Protocol