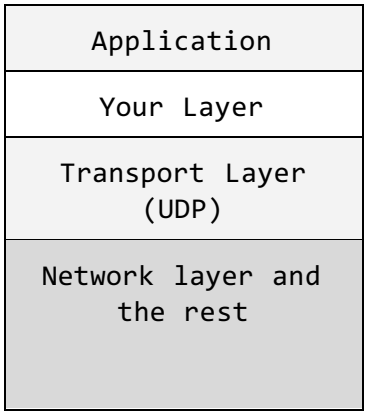
Computer Networks
Reliable Data Transfer (rdt) over UDP - Go-Back-N and Selective Repeat

UDP (User Datagram Protocol) is a simple Transport Layer Protocol. It does not have its own reliability assurance mechanism. It also is not connection oriented (No TCP Like 3 way handshake for connection establishment). Basically it does not provide any significant service over the Network Layer other than just adding source and destination port. In this project you will add reliability using Go-Back-N and Selective Repeat mechanisms.

1. Main Goal:

Add support for reliability to application layer based on the User Datagram Protocol (UDP) at transport layer. If we consider protocol stack of the internet, you will be inserting a new layer between Application and Transport layer.



2. Environment:

For this project you need to run the project in the Virtual Machine listed below. Download it from following link. Remember to run this VM you will need Virtual Box.

```
VM Link: https://utdallas.box.com/s/xwumhvregbo2jstp46cvqv3yq2s8t8jl
Virtual Box Link: https://www.virtualbox.org/
```

Inside the vm you will have a network environment where packet gets corrupted, delayed, dropped, duplicated and reordered (even with the loopback addresses and you will be using those.). Thanks to the `netem` package of linux. It helps to emulate the network properties (i.e delay, lossness) at the network level.

In this project, you will be using UDP Packets to send your data, a good first step will be to write code for sender and receiver, and send large number of dummy packets (say 1000) and see how many packets actually reaches the receiver. (In our case, it was 830, with rate of loss set to 20%)

3. Required Tasks:

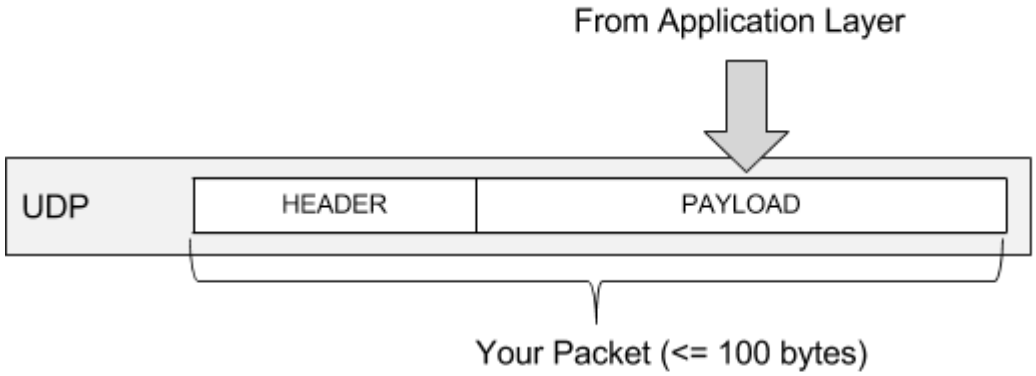
As part of this project you will need to do the following -

- Prepare a document listing all the header fields you are using (and not using) w.r.t regular TCP. In case of a field of TCP is not included in the header, give reasoning for that.
- Implement the sliding window protocol
    - Go-Back-N
    - Selective Repeat
- Implement the acknowledgement policies
    - Individual Acknowledgement
    - Cumulative Acknowledgement
- Implement policies for window size
    - Fixed Window SIze (i.e 4, 32, 64)

4. Requirements/Restrictions:

- The payload to the UDP should not be more than 100 bytes.
- Receiver should be able to detect single bit error.

Your packet structure should be as follows -



5. Code Outline:

Below you will get a rough idea how to code for solving the problem. Assume receiver is running on 127.0.0.1:2345.

Sender

```
send():
      data=get_data_to_send()
      #calculate header
      payload=header||data
      udp_send("127.0.0.1", 2345, payload)

receive_ack():
    ack=udp_receive()
    #process acks, your logic
```

Receiver

```
receive():
      payload=udp_receive()
      #your logic
      send_ack(ack_no)
```

`get_data_to_send()` will provide either new data from application layer or data for retransmission.

6. Evaluation:

Every TCP (or TCP-like) implementation focuses heavily on increasing throughput. For this project, we will be using time based measurement. *"How fast our data is transmitted using your protocol?"* . A sample text file (size ~1MB) will be provided. Your code needs to do the following -

1. Transfer the file correctly from sender to receiver.
2. Transfer the file as fast as you can.

7. What to Turn In

1. A proposed action plan by **July 2nd**.
2. A Team Report by August 6th containing
    a. Source Codes related to sender and receiver. Include a README file that describes how to compile/run the program.
    b. Project Documentations. (i.e Protocol State Diagrams, Sequence Diagram or any other document that helps to understand what has been done)
    c. Report on Header Selection
    d. 2-D plots showing how much time it requires to send a file of size 1MB using different window size and different sliding window policies.
    e. Describing what issues, if any, the team encountered during the project, how the team overcame the issues and what the team learned from the project. You can also provide suggestions on how the projects in Computer Networks could be improved in the future.
3. Individual reports, one for each team member by **August 6th**. The individual report is confidential and not shared with the other team members
    a. If you, as an individual team member, have anything specific to add to 1.e) in the team report, please do it in your individual report. Describe what issues, if any, you, as an individual team member, encountered during the project, how you overcame the issues and what you learned from the project (this is not necessarily just about the topic, could be related to teamwork, etc.). You can also provide suggestions on how

the projects in Computer Networks could be improved in the future. This complements the team report with any individual viewpoint not included in the team report.

b. Describe what each team member (including yourself) did and contributed to the project, and assign a numerical score from 1 to 10 to each team member, including yourself. 1 is the poorest, and 10 is the best.

Note: There will be a separate session (taking place outside of lecture hours) for you to demo your code is running. The research paper presentations and the code demo sessions will take place towards the end of the semester