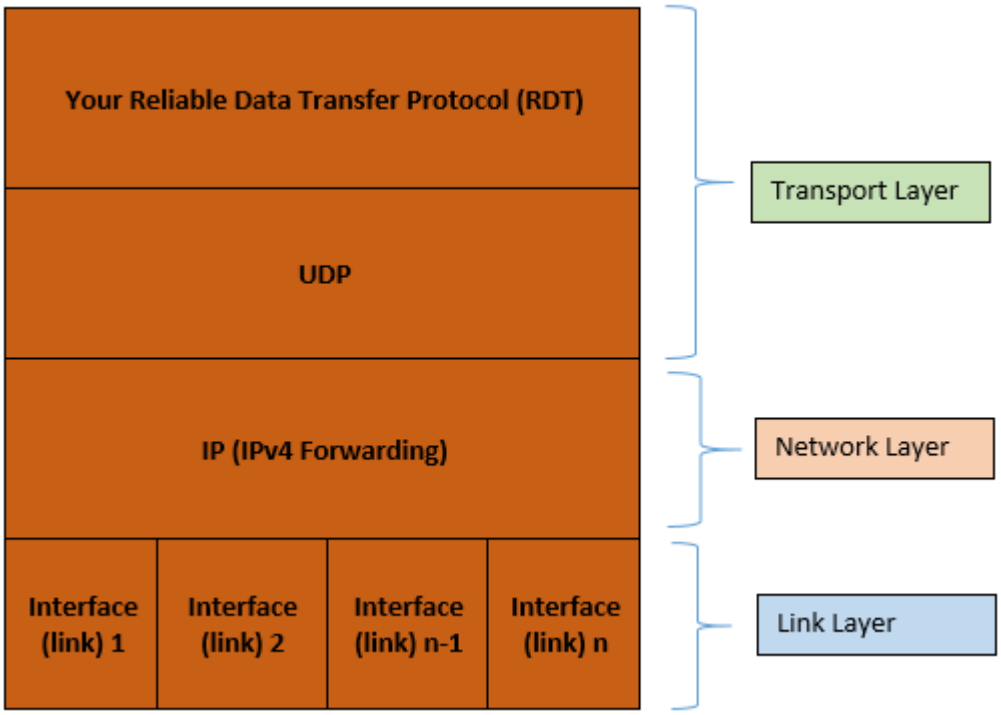
In this homework, you are expected to implement a **UDP-based** "**Reliable Data Transfer**" (**RDT**) protocol of **your own (not same as RDT 3.0 etc. in the textbook and literature)** that supports **pipelining and multi-homing**. You will also be expected to **change your network configuration**, in other words, you should modify the routing table of the nodes. Note that your protocol should be implemented by considering your own approach and design. You will concentrate on the specified topology connecting two edges and one broker (**s, B**, and **d**) over disjoint paths. The **source** will be the host **s,** and the **destination** will be the host **d**.




# 1. Network Configuration

You should be familiar with the **IPv4 addressing and routing mechanism**. You have a source and a destination node in this topology. The other nodes (i.e., routers) should forward the packets to the next hop. You have already done this work by implementing an application in the first assignment. However, you are now expected to make this configuration by modifying the routing tables of each node, namely, routing in the network layer. Please look at the "route" command for the details, and IPV4 routing mechanism implemented in the kernel.

For Instance: When a packet comes to **r1** from **B**, **r1** will forward the packet to **d**. Otherwise, if a control packet (ACK/NACK) comes from **d** to **r1**, **r1** will forward it to **B**.

For this homework, you cannot use any routing script in the routers since they will route the packets to the destination with the routing mechanism.

Moreover, you can give directly the destination IP address as a parameter to your source application. You will run your scripts only in the source, broker B and destination d. You have three terminals for running your codes one of them is the source side, the second one in the broker side, and the other one is the destination side.

# 2. Specifications

- You will implement and develop your own RDT protocol on top UDP, Thus, in this part of the project instead of UDP, you will use your own reliable protocol to send a large file from s to d. In other words, the unreliable part of the network will then be reliable. Then, the communications will be between TCP and your RDT protocol.
- Now, the broker should handle the links in a reliable fashion.
- Develop your reliable transport protocol that supports multi-homing by using UDP sockets. The above figure shows the stack including on which layer you do your implementation.
- Develop a packet-based protocol.
- You can use any programming language to implement your code, as soon as you explain the usage of it.
- Exploit disjoint links between the broker and destination. We will use the advantage of two links while transfering the file, exploiting these multiple paths will allow us to transfer the file faster than using one path. This part will provide us a multihomed protocol that you are expected to implement.
- Your source will send this large file (exactly 5 MBytes) to the destination.
- The file will be protected with a **checksum** in the assessment. In other words, the input file at the source side, and the transmitted file at the destination side should be exactly the same.
- Design your packet structure that will go into the UDP payload. **The maximum packet sizes of your RDT protocol (header + payload) can be at most 1000 bytes.**
- **Notice that, your protocol should be reusable for any topology or any network configuration based bandwidth,** delay**, and packet size. Show these details in your codes with your comments. Here, your codes will be graded based on only above topology.**
- Your codes must be **executed your source code on host s, and your destination code on host d. Any application for the routers will not be accepted. The aim of the routers is that they just route the packet to the destination and source by means of using routing mechanism (ipv4 forwarding and routing table, not applications).**
- **Run your source code on host s, and your destination code on host d:** You are expected to produce the folowing figures by changing the link properties by using "tc" command. You will change the loss, corruption and reordering property of the links explained below.
- For the usage of 'tc' command: [ http://lartc.org/manpages/tc.txt ] [ https://wiki.linuxfoundation.org/networking/netem ]

>>> You are expected to plot the following figures by using your own reliable protocol:

1. x-axis: **packet loss percentage (0.5%, 10%, 20%)**, y-axis: **file transfer time** with 95% confidence intervals. Keep the delay the same as 3 ms. Packet loss will be applied for all links. Notice that initial configurations should also be applied explained below. In this figure, the results of your RDT protocol should be represented as a line chart or bar graph. You will present your results with considering all experiments.

   `Initial Configurations for Figure 1:`**`(Please apply it for each corresponded experiment for all links between B and d)`**

   ```
   tc qdisc change dev [INTERFACE] root netem loss 0.5% corrupt 0% duplicate 0% delay 3 ms reorder 0% 0%
   ```

```
tc qdisc change dev [INTERFACE] root netem loss 10% corrupt 0% duplicate
0% delay 3 ms reorder 0% 0%

tc qdisc change dev [INTERFACE] root netem loss 20% corrupt 0% duplicate
0% delay 3 ms reorder 0% 0%
```

2. x-axis: **corruption percentage (0.2%, 10%, 20%)**, y-axis: **file transfer time** with 95% confidence intervals. Keep the delay the same as 3 ms. Packet loss will be the same for all links. Notice that initial configurations should also be applied explained below. In this figure: The results of your RDT protocol should be represented as a line chart. You will present your results with considering all experiments.

   ```
   Initial Configurations for Figure 2:
   ```

   **(Please apply it for each corresponded experiment for all links between B and d)**

   ```
   tc qdisc change dev [INTERFACE] root netem loss 0% corrupt 0.2% duplicate
   0% delay 3 ms reorder 0% 0%

   tc qdisc change dev [INTERFACE] root netem loss 0% corrupt 10% duplicate
   0% delay 3 ms reorder 0% 0%

   tc qdisc change dev [INTERFACE] root netem loss 0% corrupt 20% duplicate
   0% delay 3 ms reorder 0% 0%
   ```

3. x-axis:**reordering percentage (1%, 10%, 35%)**, y-axis: **file transfer time** with 95% confidence intervals. Keep the delay the same as 3 ms. Packet loss will be the same for all links. Notice that initial configurations should also be applied explained below. In this figure: The results of your RDT protocol should be represented as a line chart. You will present your results with considering all experiments.

   ```
   Initial Configurations for Figure 3:
   ```

   **(Please apply it for each corresponded experiment for all links between B and d)**

   ```
   tc qdisc change dev [INTERFACE] root netem loss 0% corrupt 0% duplicate 0%
   delay 3 ms reorder 1% 50%

   tc qdisc change dev [INTERFACE] root netem loss 0% corrupt 0% duplicate 0%
   delay 3 ms reorder 10% 50%

   tc qdisc change dev [INTERFACE] root netem loss 0% corrupt 0% duplicate 0%
   delay 3 ms reorder 35% 50%
   ```

Notice that, for plotting the **confidence interva**l you will have to repeat each experiment multiple times, say *n*. The margin of error less must be than 2.5%. The z-score is 1.96 for 95% confidence. We expect that the standard error (deviation) will become smaller as you increase *n*. You will have to compute the value of *n* in your experiments based on the standard error you achieve.