World building:

I am Samarth and I am sending a message to Rahul in Germany. I am using Discord. As soon as the 'send' button is pressed, the first stage, i.e., Application Layer of the process takes place…

Application Layer:

So now my computer needs to know where is Rahul's computer so that it can send the message, but for that it will need to know what server Rahul is using (Discord server in Germany). So, my computer will have to know the destination domain (discord.com), but it needs something specific, like an address, so it will have to be translated (or mapped) to machine readable address (IP address) which is unique. So, the next step that happens is called DNS Lookup which is basically my computer looking into the book of addresses to map discord.com to IP address. This mapping is what is called as **DNS Resolution.** Now this IP address can be used by routers to find the German server.

Transport Layer:

Now that we have the DNS request, we need to send data (requesting for the IP address). This message now will be sent in the form of packets (tiny and large number of packets) … basically data divided into numerous tiny packets. There are mainly two types of protocols: **TCP** and **UDP**, which "prepare the packets" … for the DNS query, fast and not-so-strict protocol can be used, which is the **UDP** (this might lead the packet being lost, but is faster for a small request like this). Now that the DNS query is sent, DNS resolution takes place…

(so far only data sent is to find the IP address of the domain, which is discord.com)

> **DNS Resolution:** This is the process of basically mapping the human readable domain name (discord.com) to IP address. This happens, when the client (my computer) asks my configured DNS resolver (my Internet Service provider i.e., the company that I pay my internet bills) "hey, what is the IP address of this umm… di-di-dis- discord.com?". Then the ISP responds and provides the IP address needed and then now we will have to send the actual data (the message).
>
> So far, we have only got the IP address of the discord server in Germany. Earlier, the protocol used to send the DNS query was UDP, which was fast and careless, but now we are sending a far more valuable data than a DNS query, i.e., the message itself… so we need to be careful that the message is sent in order and in full… for this we use a much stricter protocol **TCP** to prepare the packets…

Now TCP uses what is called a **three-way-handshake** process to establish a connection between the two sides.

Three-way-handshake: This mainly contains three steps of connection establishment between the two sides.
SYN(chronize) - used to initiate a connection, sent by the client.
SYN(chronize) ACK(nowledgement) - the server sends back packets to accept the connection.
ACK(nowledgement) - sent by the client again to server to confirm the connection

Now, TCP does two more critical tasks for the message:

Segmentation: the message is broken down into smaller pieces called segments, which is necessary for efficient transfer across different network hardware…

Port addressing: A TCP header is added to each segment which includes **Source Port** (a random port number on my computer) and **Destination Port** (port number used by discord application in German server)**.**

Network Layer: Global addressing and routing

The basic job of this network layer is to get the packet from its source to its destination (from my computer to discord server in Germany). For this, two essential components are added to the packet – **Source IP address** (my computer's public address) and **Destination IP address** (address of discord server in Germany)**.**

BUT HOW?

The packets need to travel a great distance to Germany… that's why at different stages of these distances; different routings are followed. (Kind of like, if you have to go to Germany, first you will have to go to Mangalore airport first via car or bus, then walk till the aeroplane, then go to Germany in flight (if there is a direct flight) or else take multiple flights each stopping at various locations between Mangalore and Germany).

This is mainly because there is limit up to the area my local internet service provider can cover – maybe whole of Mangalore is covered by the internet which the company gives me – what will the packets do after Mangalore? – that's the main question…

While in Mangalore (Routing inside ISP): The entire area covered by my ISP is a single administrative entity called **Autonomous System (AS).** This is really easy and fast as these internal **routers** use efficient algorithms to find the best internal path to get the packets outside Mangalore and toward the regional exchange point.

Outside Mangalore (Across autonomous systems): After the range of my local ISP ends, the **BGP (Border Gateway Protocol)** takes over. BGP is the main routing protocol of the global

internet.
How BGP works? => it can be thought of as a huge map where each AS advertises which IP addresses it can reach. The routers then “talk to each other” to decide which neighbouring AS offers best path to Germany. The packet hops from one major network to another, even travelling through undersea fiber optic cables.

NAT (Network Address Translation) - My PC would likely use a private IP address which I wouldn’t want the public networks accessing my data packets to know about; thus, ISP’s gateway performs NAT, which basically translates(replaces) my private Source IP Address with ISP-assigned public IP address before packet leaves India, and this is what the German network would see…

Data Link Layer and Physical Layer:

So now that the packets (message) have reached the German discord server (local network segment), the **Data Link layer** takes over. The packets go through **Encapsulation/Framing,** where the packets are wrapped in a frame (like an ethernet frame), this frame mainly adds the **MAC (Media Access Control) address** necessary for local delivery. MAC address is a unique address on a hardware device that helps the devices to share information on a local network. The frame contains MAC addresses of the Source (the last router) and the Destination (the discord server’s Network Interface Card (NIC)). From this stage, after framing, it goes to Physical Layer.

The **Physical Layer** is the actual transmission medium. It takes the data from the Frame and converts it into a signal that can travel across the hardware. The final short distance (within the German Data centre) is sent over Ethernet cables. After the signal reaches the Discord server’s NIC, the process is reversed and the signal is converted back to the frame, and the frame is stripped to reveal the IP packet and passes the packet to Network layer on the server’s operating system.

Now that the German discord server has received my message, it will again follow the same steps to send the message from the server to Rahul across various Autonomous Systems mentioned earlier…

The Reverse Path: From Germany to India

Now that the server has received the message, it would send a confirmation message back to my pc (ACK packet). It prepares the packets and sends it over the internet and follows the same path again but the source and destination addresses are now reversed.

NAT Reversal: As my ISP had previously translated my private IP to public one, it should be translated back, so that it reaches my PC. My ISP's router sees a packet destined for my

public IP address on a specific port. It consults its **NAT table** and realizes this packet belongs to the connection I initiated.

Effect of Congestion:

If one of the intermediate routers in the journey is overloaded, it will slow down increasing the **Round-Trip Time (RTT),** this means that packet arrives late, delaying next segment of data flow.

Effect of Packet Loss:

If a router's buffer overflows due to severe congestion, it may drop a TCP segment. When this happens, the receiver tells the sender that there is a gap in the sequence numbers and that a segment is missing, the sender must then retransmit the lost segment.