# Class Notes

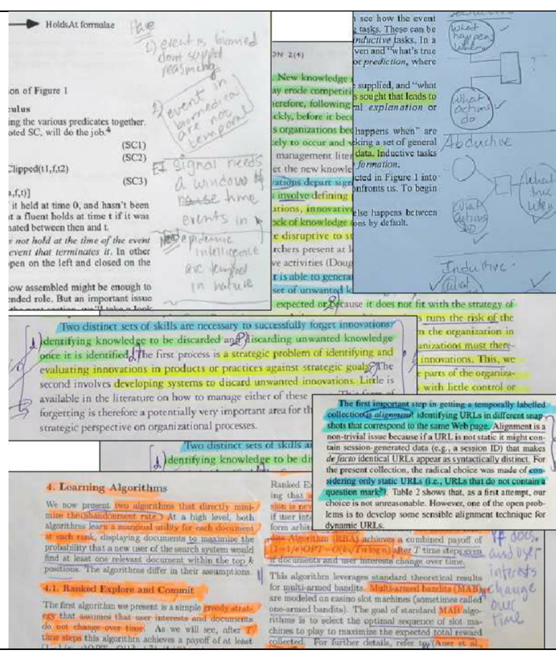
## Presentation 1:

You should be done with your presentation 1 latest by **23rd November 2025 11:59 midnight**.

## Presentation 2: Falcon: A Reliable, Low Latency Hardware Transport

We have selected Falcon paper for your presentation 2.
This paper was presented at ACM's SigComm 2025 conference.
SigComm is an elite venue for novel networking research.
Paper is available at: https://dl.acm.org/doi/pdf/10.1145/3718958.3754353
Deadline for you to thoroughly read the paper: **November 30th, 2025**
If we had time, we would have 1 minute lightening talks in the class.
Else, we will use a quiz to verify that you understand what the paper says.
You also need to heavily annotate the paper.

- You should write any term you are not aware of on the paper.
- You can use any AI tool to help you understand the paper. You can upload the paper to AI tool for this purpose as well.
- You should know:
    - What problem paper is solving (need)
    - How they solved the problem (approach)
    - What benefits their solution will provide (benefits)
    - What other ways are there to solve the same problem. Cons of their approach. (competition)
- Following is an example how annotations look like:



You will also need to upload your annotated paper to GCR.

# Network Address Translation (NAT)

Abandoning classful IPv4 IP allocation scheme for more flexible CIDR-based scheme, slowed the waste of IPv4 addresses by matching the allocation to demand as much as possible (there can be some waste still due to our need to allocate in power of 2), and CIDR helped network prefix summarization so that entries in routers' forwarding tables grow slowly (very important if a router has to do its job at line-rate).

But CIDR-based scheme did not solve the IPv4 scarcity problem.
Internet clearly needed a larger address space.
While IPv6 was yet not ready, a stop-gap scheme called NAT was introduced.
NAT was successful to substantially deal with the IPv4 scarcity and companies did heavy investments in this technology.
But this success comes at a heavy cost! NAT boxes change IP and Transport layer headers and that goes against Internet's original principle that the core of the network has no business changing packets and should not look beyond network layer.
Unfortunately, we are stuck with NAT even when IPv6 was fully ready to go.

Over to slides (slide no 63) to learn about NAT

## Carrier-Grade NAT (CGNAT) – Next level ugliness!

See the following trace:

```
traceroute to www.google.com (142.250.200.164), 64 hops max, 40 byte
packets
 1  192.168.2.1 (192.168.2.1)  7.195 ms  3.195 ms  7.155 ms
 2  * * *
 3  10.253.10.193 (10.253.10.193)  40.821 ms  36.049 ms  39.711 ms
 4  10.253.9.146 (10.253.9.146)  120.253 ms
    10.253.9.216 (10.253.9.216)  58.037 ms
    10.253.9.209 (10.253.9.209)  37.745 ms
 5  * * *
 6  10.253.10.1 (10.253.10.1)  105.865 ms
    10.253.10.65 (10.253.10.65)  30.173 ms
    10.253.10.1 (10.253.10.1)  41.161 ms
 7  10.253.14.253 (10.253.14.253)  124.622 ms
    10.253.14.252 (10.253.14.252)  40.732 ms  37.528 ms
 8  10.253.0.190 (10.253.0.190)  60.252 ms
    10.253.0.186 (10.253.0.186)  53.085 ms  56.004 ms
 9  static-host119-30-106-58.link.net.pk (119.30.106.58)  46.545 ms
32.165 ms  44.580 ms
```

```
10  static-host119-30-106-57.link.net.pk (119.30.106.57)  39.726 ms
51.302 ms  43.640 ms
11  110.93.249.184 (110.93.249.184)  194.104 ms  34.595 ms  62.581
ms
12  110.93.253.226 (110.93.253.226)  86.862 ms
    110.93.255.127 (110.93.255.127)  41.584 ms
    110.93.255.26 (110.93.255.26)  55.049 ms
13  110.93.252.146 (110.93.252.146)  59.974 ms
    110.93.252.136 (110.93.252.136)  82.041 ms
    110.93.252.198 (110.93.252.198)  41.573 ms
14  110.93.192.109 (110.93.192.109)  96.483 ms
    110.93.192.111 (110.93.192.111)  75.656 ms
    110.93.252.136 (110.93.252.136)  60.270 ms
15  192.178.252.233 (192.178.252.233)  101.791 ms *
    110.93.192.109 (110.93.192.109)  62.634 ms
16  72.14.239.47 (72.14.239.47)  62.953 ms
    192.178.87.254 (192.178.87.254)  68.421 ms *
17  pnfjra-aq-in-f4.1e100.net (142.250.200.164)  104.263 ms
    192.178.96.8 (192.178.96.8)  79.357 ms
    74.125.37.203 (74.125.37.203)  64.516 ms
```

## IPv6 – 128 bit address space!

Huge address space: 2^128 = 2^32 * 2^32 * 2^32 * 2^32
We write IPv6 addresses in 16 bit groups in hexadecimal. Total groups = 128/16 = 8 groups
**Example:** `2404:3100:1829:14d:c52b:4d3f:3b3e:4352`

### IPv6 compaction rules

| Rule | Description | Correct Example | Incorrect Example (Non-Example) |
|---|---|---|---|
| 1. Omit Leading Zeros | In any 16-bit block (hextet), you can remove the zeros from the beginning of the block. | Full: 2001:0db8:0001:000a:8a2e:0370:7334<br>Compact: 2001:db8:1:a:8a2e:370:7334<br><br>A block of 0000 becomes 0. | Address: ...:8a00:...<br>Incorrect: ...:8a:... ❌<br>(You cannot remove trailing zeros, as this changes the value of the hextet). |
| 2. Compress Consecutive Zeros | You can replace one contiguous group of all-zero hextets with a double colon (::). | Full: fe80:0000:0000:0000:0202:b3ff:fe1e:8329<br>Compact: fe80::202:b3ff:fe1e:8329<br>(The :: replaces the three 0000 blocks).<br><br>Loopback: 0:0:0:0:0:0:0:1<br>Compact: ::1 | Address: 2001:0:0:1:0:0:0:1<br>Incorrect: 2001::1::1 ❌<br>(You can only use the :: once. Using it twice makes the address ambiguous). |
| 3. Compressing the Right Group | If you have multiple groups of zeros, you must compress the longest one.<br><br>If groups are of equal length, compress the first (leftmost) one. | Longest: 2001:0:0:1:0:0:0:1<br>(Groups are 0:0 and 0:0:0).<br>Correct: 2001:0:0:1::1 ✅<br><br>Equal: 2001:0:0:1:0:0:1:1<br>(Groups are 0:0 and 0:0).<br>Correct: 2001::1:0:0:1:1 ✅ | Longest: 2001:0:0:1:0:0:0:1<br>Incorrect: 2001::1:0:0:0:1 ❌<br>(This compressed the shorter group).<br><br>Equal: 2001:0:0:1:0:0:1:1<br>Incorrect: 2001:0:0:1::1:1 ❌<br>(This compressed the second group). |
| 4. Don't Compress a Single 0 | Per RFC 5952 (the standard for text representation), you should not use :: to compress a single hextet of zeros. | Address: 2001:db8:1:a:0:8a2e:370:7334<br>Correct: 2001:db8:1:a:0:8a2e:370:7334 ✅<br>(You just leave the single 0). | Address:<br>2001:db8:1:a:0:8a2e:370:7334<br>Incorrect:<br>2001:db8:1:a::8a2e:370:7334 ❌<br>(This is not the preferred canonical format). |

(Source: Google Gemini)

```
In theory, all the CIDR rules we learned for IPv4 apples to IPv4.
In practice, much larger blocks are given out as a norm because we have
no shortage of IPv6 addresses.
Typical IPv6 allocation sizes:
```

| Level | Typical CIDR Block | Recipient / Purpose |
|---|---|---|
| **IANA** | **/12** | **RIRs** (Regional Internet Registries like ARIN, RIPE, APNIC) |
| **RIR** | **/32** | **ISPs** (Local Internet Registries). This is the standard *minimum* allocation for an ISP. |
| **ISP** | **/48** | **Business / Enterprise Customer**. This gives the business 65,536 individual `/64` subnets. |
| **ISP** | **/56** | **Home / Residential Customer**. This gives the home 256 individual `/64` subnets (for "Guest", "IoT", "Main", etc.). |
| **End User** | **/64** | **A Single Subnet**. This is the standard, fundamental size for one LAN (e.g., your Wi-Fi network). |

Source: Google Gemini

More on IPv6: Go to slide no 69

## Dynamic Host Configuration Protocol (DHCP)

Question: How does an interface get an IP?
Answer: Usually using DHCP protocol.

Go to slide no 51

## Classic vs SDN way of Control Plane

Head over to slide no 78