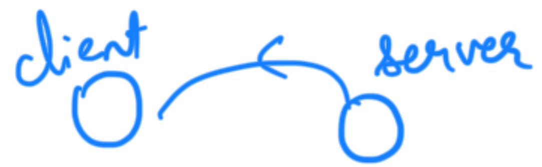
CS348 Notes
P2P Networks (Application Layer)
Video Numbers: 28, 29

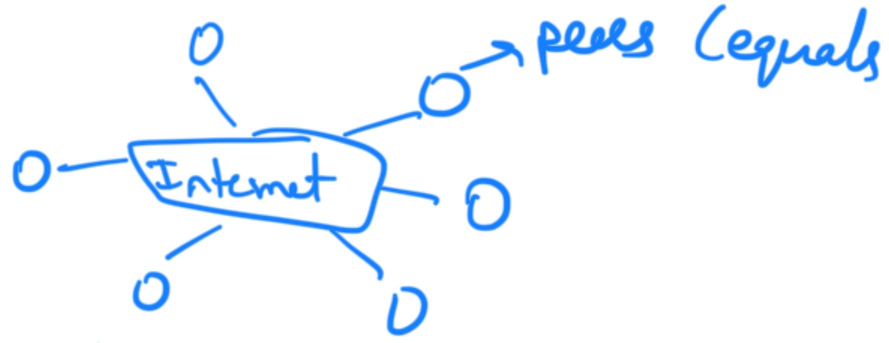
OjMaha

I have prepared these notes by watching the videos from Networks Playlist. The following notes may be asynchronous and irrelevant to what Prof. Vinay teaches in class (cuz I do not pay attention during lectures lol). Further, these notes might not cover *everything* as explained in the video lectures. Consider these to be a supplemental read :). If you find any errors, do notify me so they can be edited.

APPLICATION LAYER

P2P Network : (peer-to-peer networks)





Or2 company Napster was 1st to use P2P network to share music.
Essentially if peer1 bought Bruno Mars; peer2 bought Coldplay; they can share their music with each other over the Napster network (server).

So the napster server has the following info :

File1	IP_A
File2	IP_B

i.e. A has file1 & B has file2.

Now C logs in to the server and searches for song; gets a hit on file1.

Drawbacks : i) centralised (if server down then ded)

: ii) legally ez to take down. Napster lost case & shut down.

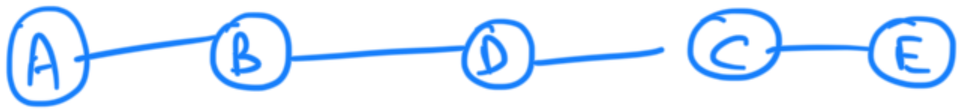
G-Nutella :

Bootstrapping : i) Applicⁿ may have IPs of some other peers.

ii) Or it has mechanism to lookup for IPs from websites.

↓

to set up the peer network (how will a new peer join?)



Say A wants to search for a file "f" but B doesn't have it. So B forwards (broadcasts) the query ahead. But we can't keep broadcasting as it seems overkill. So we do a limited broadcast. (Have a ttl & broadcast only till a 2-3 hops away)

Say n=2 & D has the file "f". How does A know abt it?

Method 1: The Query has "f" & a "QID" (unique query id) (no IP_A)

First, B receives (QID, A) meaning A has requested for query with QID.

Then, D receives (QID, B). D has file with QID. So it replies back to B.
B had cached the (QID, A) request. Then B replies to A.

D replies to B as follows: (QID, IP_D, $Port_D$)

B also checks its cache entry (QID, A) and replies to A with (QID, IP_D, $port_D$).

Now, A can download from its peer "D". Reply is on the path traversed here.

B stores the reply (QID, IP_D, $port_D$) and caches it. So next time when say "G" asks B for file "f"; then B doesn't fwd request but instead simply shares the prev. cached info.

If you don't find file within the ttl; u can inc. till max ttl & try.

Method 2: query has IP_A. Reply directly to A.

Ltd broadcast is $O(n)$. Can we do it in $O(\log n)$??
$\hookrightarrow$ no. of nodes.

So, we have IP_A ..... IP_Z.

say we [illegible]

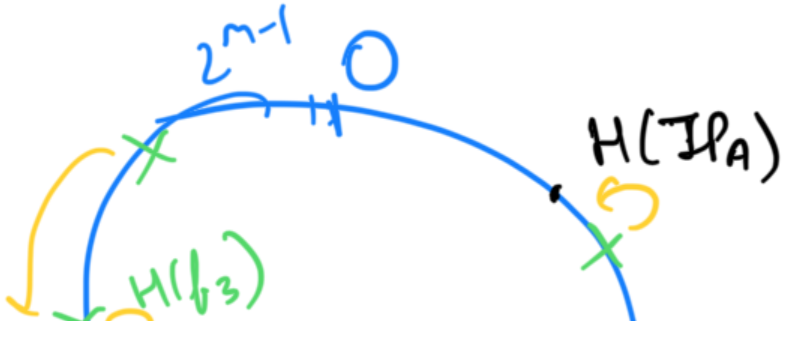
IP_Q has f_1. IP_S has f_2. Suppose Q tells E, S tells B that they have f_1 & f_2 respectively. Say A wants f_1. And say A knows somehow he needs to contact E. Then problem solved as E will say Q has f_1.

We just need some sort of mapping from IP addresses to files. This mapping scheme would be known across all IPs so when querying, one knows whom to contact.

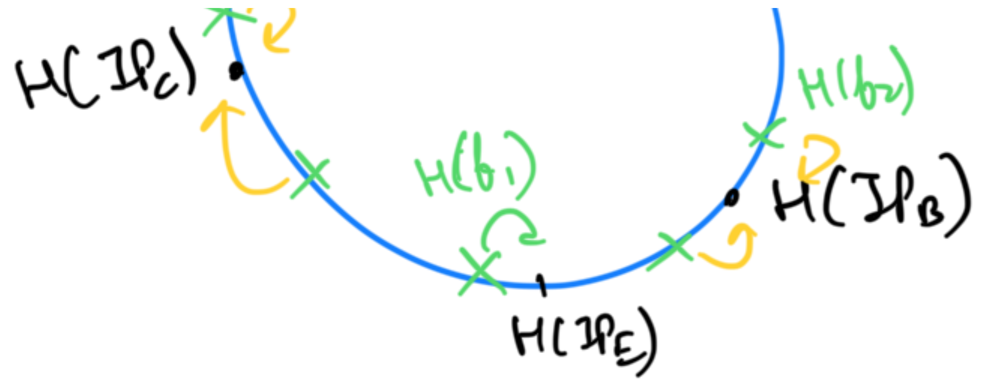
For the mapping; use HASH function.

$$f_1 \text{ (of any length)} \longrightarrow H(f_1) \text{ of length 'm' bits.}$$

But, even if 'H' generates uniform IPs, it is possible that the available IPs themselves are not uniformly distributed.



X : hashes of files.



Note that H(f) are uniform. But; H(IP) isn't. Thus, IP_c ends up doing most work.

Say Q has f_1. It computes H(f_1). Now, it needs to find out the closest node to H(f_1) i.e. E and tell it that Q has f_1. Further, what if Q_1 leaves the network? What to do then?