Application Layer

Lecture 28 Peer-to-Peer Networks

In the client-server model, the client gets a service from the server.

In a P2P network, there are many equal peers, who can provide services to each other.

→ How do we start using it? What services? How do we provide services?

It arose from the desire to share digital content, so not everyone needs to pay.

Napster had a central server, and logged in people could upload what files they have and freely share them. Others can search the server for some song, and download it from someone else if there is a match.

It finally shut down (it was illegal, after all).

- Centralised, so not fault-tolerant
- Legally easy to take down.

After Napster came GNUTELLA. It tried to get rid of the problem of being centralized.

To join a pre-existing network (bootstrapping),

→ The application may have the IPs of some peers.

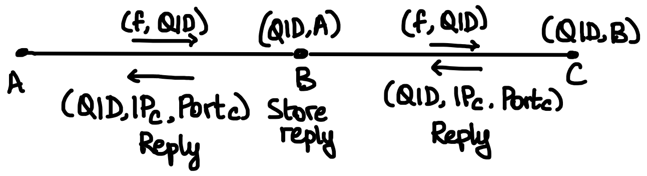
or

→ We may be able to look up IPs from one or more websites.

To search for a file,

We do a limited broadcast — a broadcast but limited to a certain number of hops away. We can do so by setting a TTL at the application layer. Suppose the query has f (file name) and the query ID (but not the requester's IP. Each host who receives this query caches the query ID and the peer they heard it from. When it reaches someone who has a file, they reply with their own IP and Port number (with the query ID). This goes backwards until the requester gets to know who has the file, so we can then share.

Further, the intermediate peers store the reply for the particular file.



At its peak, there were millions of people connected to GNUTELLA.

The QID is to ensure that we do not redundantly send the same message multiple times (if received multiple times).

In an improved version, we can also store the requester's IP to allow a direct reply.

In v0.4, TTL = 7

v0.6, TTL = 4

+ There is no centralized node.
- We frequently broadcast.

How do we fix this? Is there a way to

- map files to IP addresses such that
- the IP stores who has a particular file.

$$\begin{tikzcd}
\text{(centralized} \arrow[d] & \text{but distributed)} \arrow[d] \\
\text{a single person knows who has the particular file} & \text{this person is different for different files.}
\end{tikzcd}$$

We would like to do this pseudo-randomly ensuring that everyone has the mapping.

To do this, we use a hash function (which is deterministic).

(arbitrary length bitstring $\mapsto$ fixed length bitstring)

The hash values will be more or less evenly distributed over the space.

To map to IP addresses, we further hash the IPs.

A hash (of a file) is then mapped to the hash (of an IP) that is closest to it.

How do we do this? What do we do if a mapped IP leaves the network?