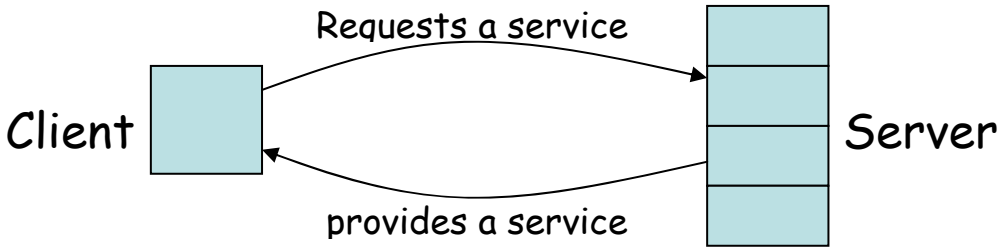
Peer-to-Peer Networks

January 14, 2008

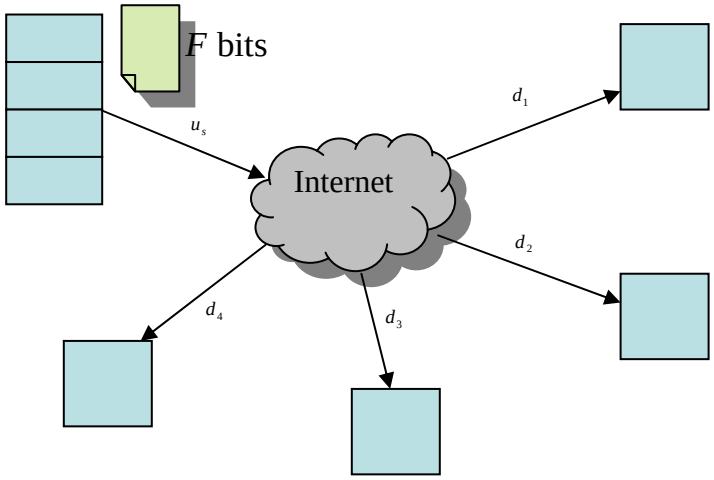
The goal of today's lecture is to introduce peer-to-peer networks, focusing on peer-to-peer file distribution/sharing.

1. Motivation

In the traditional client/server model in the Internet, each end host is either a client or a server. A client runs a client program and requests service. A server runs a server program and provides service. One outstanding example of client vs. server is Web browser vs. Web server. Clients are sometimes on and the one who initiates a request of service to the server. The server is usually always on and provides services to many clients. Clients do not communicate directly with other clients.



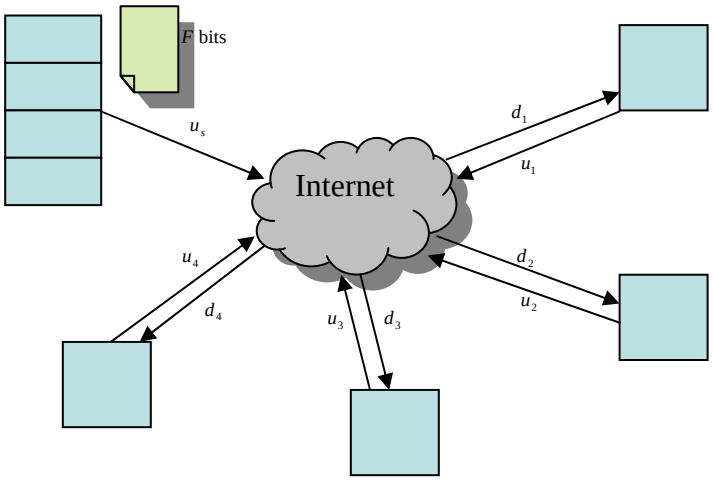
Now we consider the problem of distributing a large file by a server to N receivers (clients). Suppose that the server has an upload rate of u_s, the file size is F bits, and each receiver i has a download rate of d_i. The server needs to send NF bits, which takes at least NF/u_s time. Also, the slowest receiver takes at least $F/\min_i\{d_i\}$ time to receive the file. So the time for all the receivers to receive the file is at least $\max\{NF/u_s, F/\min_i\{d_i\}\}$.



When the number of the receivers is large, the term NF/u_s dominates and the download time scales with N. There is a number of ways to speed up the file distribution. For example, we can increase the link bandwidth at the server or use multiple servers. This, however, requires deploying more infrastructure.

1.1 Peer-to-Peer file distribution

An alternative is to let those receivers that got a copy of the file to redistribute the file to other receivers. This will reduce the burden at the server and potentially speed up the file distribution.



Suppose additionally that each receiver i has an upload rate of u_i. The total upload rate is $u_s + \sum_i u_i$, and it thus takes at least $NF/(u_s + \sum_i u_i)$ upload time. The server needs to send at least one copy of the file, which takes F/u_s time. The slowest receiver takes at least $F/\min_i\{d_i\}$ time to receive the file. So the time for all the receivers to receive the file is at least $\max\{NF/(u_s + \sum_i u_i), F/u_s, F/\min_i\{d_i\}\}$. You can see that peer-to-peer file distribution is self-scaling. It has much lower demand on sever upload capacity and the distribution time grows only slowly with the number of the receivers.

1.2 Challenges in Peer-to-Peer networks

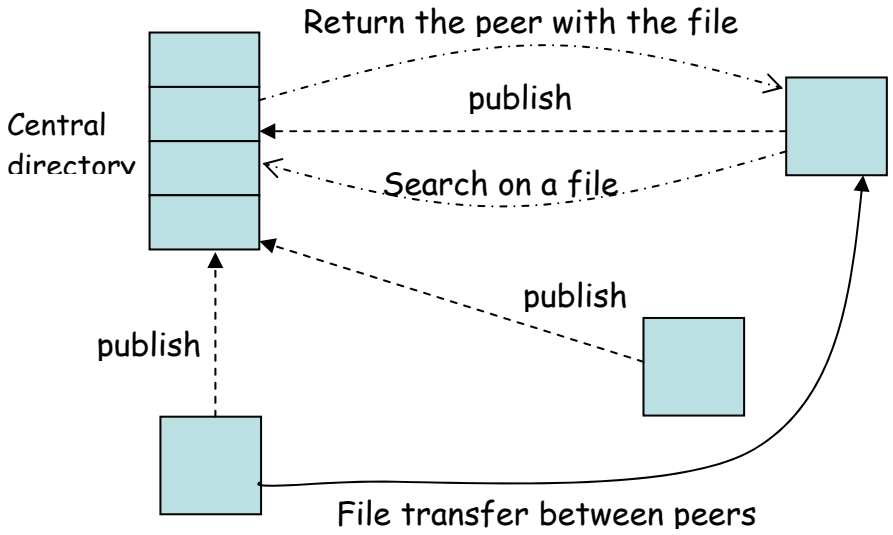
There are many challenges in peer-to-peer file distribution though. Peers may come and go. Peers may not be willing to help others. Peers need to find the peer with the content they want. So, peer-to-peer networks need mechanisms to handle time-varying connectivity, to motivate peers to contribute, and to locate the relevant peers. In this lecture we will mainly focus on search/lookup, i.e., how to locate the relevant peers.

2. Search/lookup in Peer-to-Peer networks

There are three main approaches to locate relevant peers: central directory, query flooding, and using hierarchical overlay. Each of these approaches is implemented in some popular peer-to-peer networks.

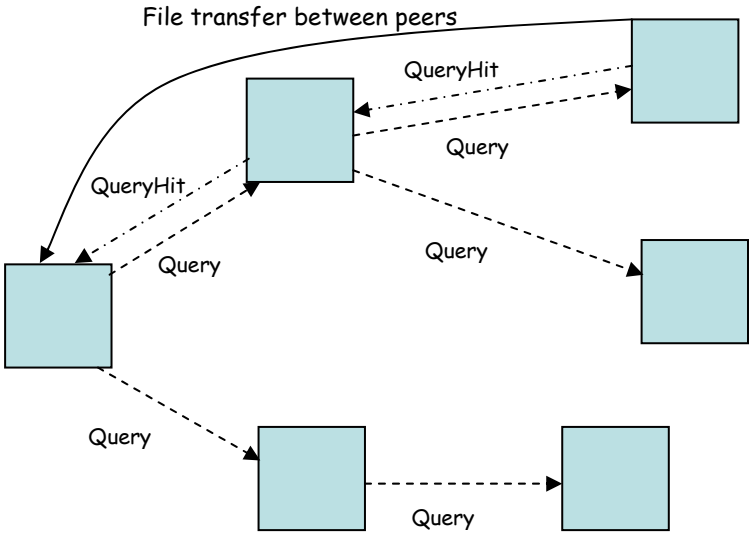
2.1 Central Directory and Napster

Directory service is implemented in early peer-to-peer networks such as Napster. Napster was first released in January 1999 and popularized the idea of peer-to-peer file distribution/sharing. In Napster, a peer will download and install a client program and contact Napster via TCP to provide a list of files it would like to share (i.e., to publish). The Napster's central server continually updates the directory. When a peer searches on a file, Napster will identify the peers with the file and provide the corresponding IP addresses. The peer will request and receive the file directly from the chosen peer. This peer-to-peer file transfer imposes no load on the server. While file transfer is decentralized, locating the file is highly centralized and has problem of single point of failure.



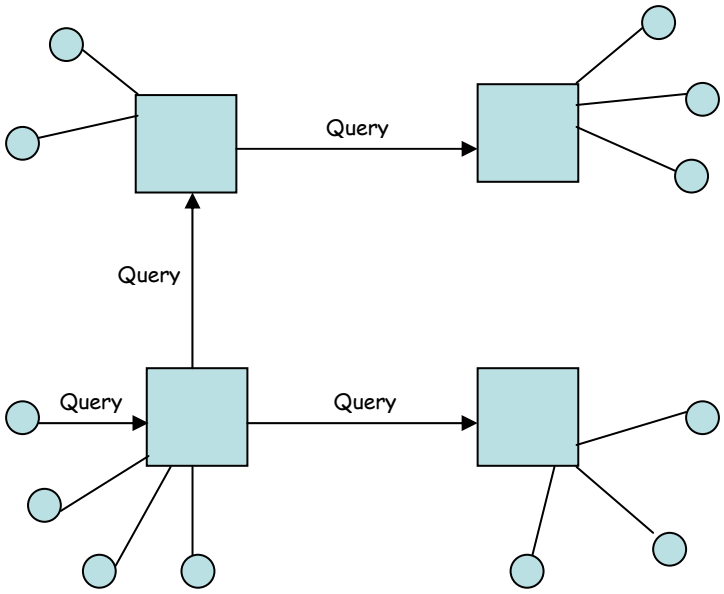
2.2 Query Flooding and Gnutella

Query flooding is used in later peer-to-peer networks such as Gnutella. In Gnutella, a peer joins the network by contacting a few nodes to become neighbors. A peer does not need to publish. When a peer wants to download a file, it will ask neighbors who will ask their neighbors and so on. When a peer has the file queried, it will send back a QueryHit over the reverse path. This flooding approach is fully decentralized and the search cost is shared among peers. But the search scope may be very large and the search time can be very long. These problems can be reduced to some extent by limiting the hop-count the query travels.



2.3 Hierarchical Overlay and KaAzA

In KaAza networks, peers constitute a hierarchical overlay network with some peers serving as super-node (group leader). When a peer joins the network, it contacts a super-node (and may later be elected as a super-node) and sends a list of files it would like to share (i.e., publish) to the super-node. The super-node keeps track of content in all its children. When a peer wants to download a file, it sends a query to its super-node, and the super-node floods queries among super-nodes. Since many peers may have only a few files, super-nodes consolidate queries and save time in searching.



3. BitTorrent Networks

So far we have focused on search but not on file fetching. BitTorrent focuses on efficient fetching. It is motivated by a situation where a single publisher has a large file that many receivers want to download at the same time (flash crowd), e.g., when the new version of a popular game is released. BitTorrent divides a large file into many small pieces, called chunks. Different chunks are replicated on different peers. This allows simultaneous downloading. A peer retrieves different parts (chunks) of a file from different peers, uploads different parts to peers, and assembles the entire file when it gets all the pieces.

BitTorrent has an infrastructure node – tracker that keeps track of peers in the torrent. A peer registers with the tracker when it arrives and periodically informs tracker when it is still there. Tracker returns a random set of peers in the network. The peer will connect the peers in this set and periodically ask them for the list of chunks they have. One of key issues is which chunk to download from peers. If the chunks are downloaded in order, then many peers have the early chunks. In this situation, peers may have little to share, which limits the scalability of the network. It is also possible that nobody has the rare chunks, which may keep peers from completing a download. Instead, BitTorrent downloads the rarest chunk first, i.e., downloads the chunk with the fewest available copies first. This avoids starvation across peers and balances load by equalizing the number of the copies of each chunks. All these will increase the utilization of bandwidth and speed up the file distribution.

In many peer-to-peer networks, vast majority of peers are free-riders that share no files and answer no queries or limit its upload speed, and a few peers essentially act as servers to contribute to public good. The free-riding behavior reduces the efficiency of the whole system and may render its collapse. BitTorrent uses a simple incentive mechanism Tit-for-Tat to prevent free riding: to upload to the peers that are uploading at highest rate. A peer measures the download rates from peers and uploads to the top four peers, and recomputes and reallocates every 10 seconds. In order for the new peers to get chance to be served, a peer will randomly upload to a new peer every 30 seconds. This is called optimistic unchoking.

BitTorrent is the most popular peer-to-peer file distribution system. It is estimated to account for 30% of the Internet traffic. Despite its huge success and popularity, BitTorrent faces several challenging problems such as the problem of incomplete downloads and the need for robust incentive mechanisms.

4. Summary

As mentioned, two of the key issues in peer-to-peer networks are search and incentive. In this lecture, we have discussed search in the context of several popular peer-to-peer networks. We also discussed BitTorrent networks that focus mainly on efficient file fetching. In the next lecture, we will discuss the incentive issue and give a brief introduction to basic concepts in game theory and mechanism design.