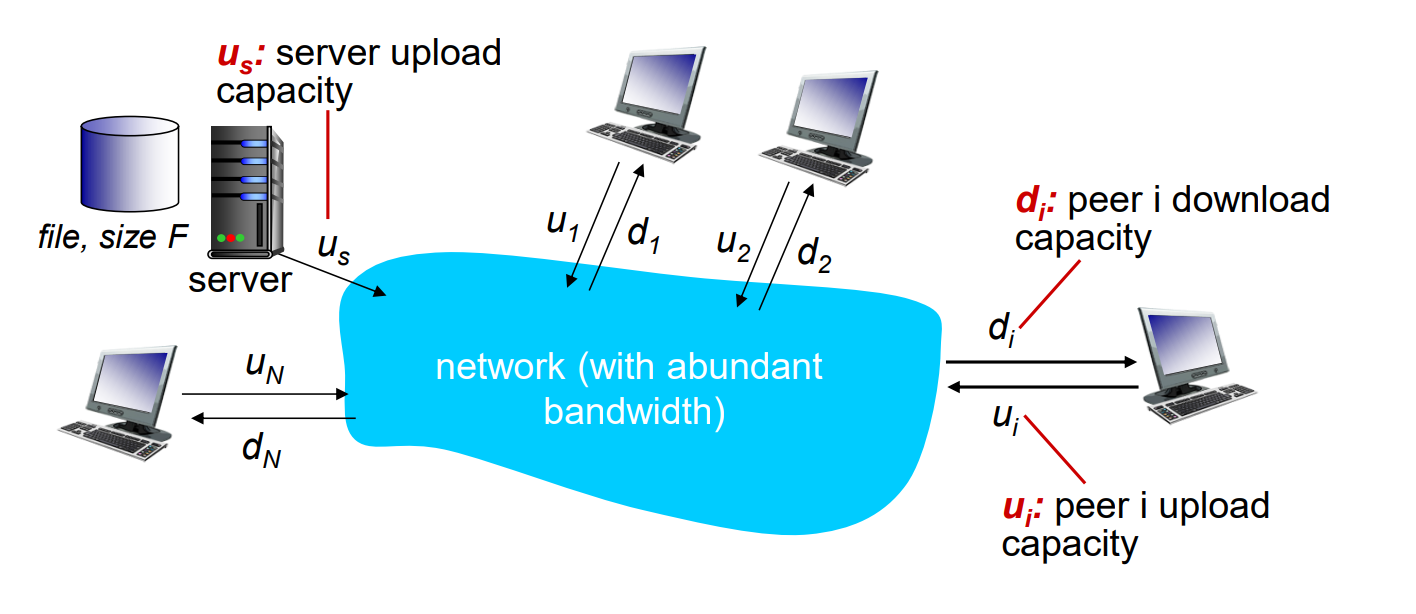
Suppose, the server and the peers are connected to the Internet with access links.   
Denote the upload rate of the server’s access link by us , the upload rate of the ith peer’s access link by ui , and the download rate of the ith peer’s access link by di .  
Also denote the size of the file to be distributed (in bits) by F and the number of peers that want to obtain a copy of the file by N.

The distribution time is the time it takes to get a copy of the file to all N peers.



SUPPOSE LARGE BANDWIDTH

Let’s first determine the distribution time for the client-server architecture, Server transmission time: which we denote by Dcs.  
In the client-server architecture, none of the peers aids in distributing the file. We make the following observations:

• The server must transmit one copy of the file to each of the N peers. Thus the server must transmit N\*F bits. Since the server’s upload rate is us , the time to distribute the file must be at least N\*F/us .

Receiving time for each client Client:   
• Let dmin denote the download rate of the peer with the lowest download rate,

that is, dmin = min{d1,dp,...,dN}.

The peer with the lowest download rate can obtain all F bits of the file in F/dmin seconds. Thus the minimum distribution time is at least F/dmin.

Putting these two observations together,  
Dcs >= max{N\*F/us , F /dmin }

Number of user , increase

the distribution time increases linearly with the number of peers N

similar analysis for the P2P architecture:

Here peer can assist the server in distributing the file. When a peer receives some file data, it can use its own upload capacity to redistribute the data to other peers.

Distribution time:

At the beginning, the server must send each bit of the file at least once into its access link. for that,  
time : F/us

Now As with the client-server architecture, the minimum distribution time is at least F/dmin.  
So, total upload capacity of the system as a whole is equal to the upload rate of the server plus the upload rates of each of the individual peers, utotal = us + u1 + … + uN

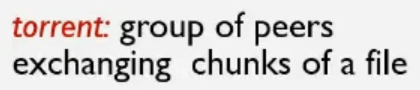
Putting these three observations together, we obtain the minimum distribution time for P2P,

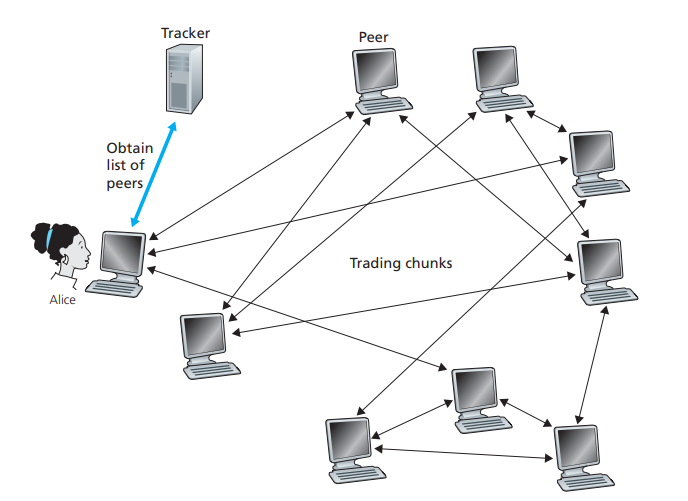
DP2P =max { F /us , F /dmin , N\*F /(us + ui )}

Real life example for p2p: (Torrent)

Peers in a torrent download equal-size chunks of the file from one another, with a typical chunk size of 256 KBytes. When a peer first joins a torrent, it has no chunks. Over time it accumulates more and more chunks. While it downloads chunks it also uploads chunks to other peers.

torrent has an infrastructure node called a tracker. When a peer joins a torrent, it registers itself with the tracker and periodically informs the tracker that it is still in the torrent. In this manner, the tracker keeps track of the peers that are participating in the torrent.





when a new peer, Alice, joins the torrent, the tracker randomly selects a subset of peers.

Possessing this list of peers, Alice attempts to establish concurrent TCP connections with all the peers. Let’s call all the peers with which Alice succeeds in establishing a TCP connection.

each peer will have a subset of chunks from the file, with different peers having different subsets. Periodically, Alice will ask each of her neighboring peers (over the TCP connections) for the list of the chunks they have. If Alice has L different neighbors, Alice will obtain L lists of chunks. With this knowledge, Alice will issue requests (again over the TCP connections) for chunks she currently does not have. Alice will have a subset of chunks and will know which chunks her neighbors have.

Alice request for rarest chunks first.  
Alice sends chunks to those four peers currently sending her chunks at highest rate.