Computer Communication Networks

Application Layer

Dr. Raja Vara Prasad

Assistant Professor

IIIT Sri City

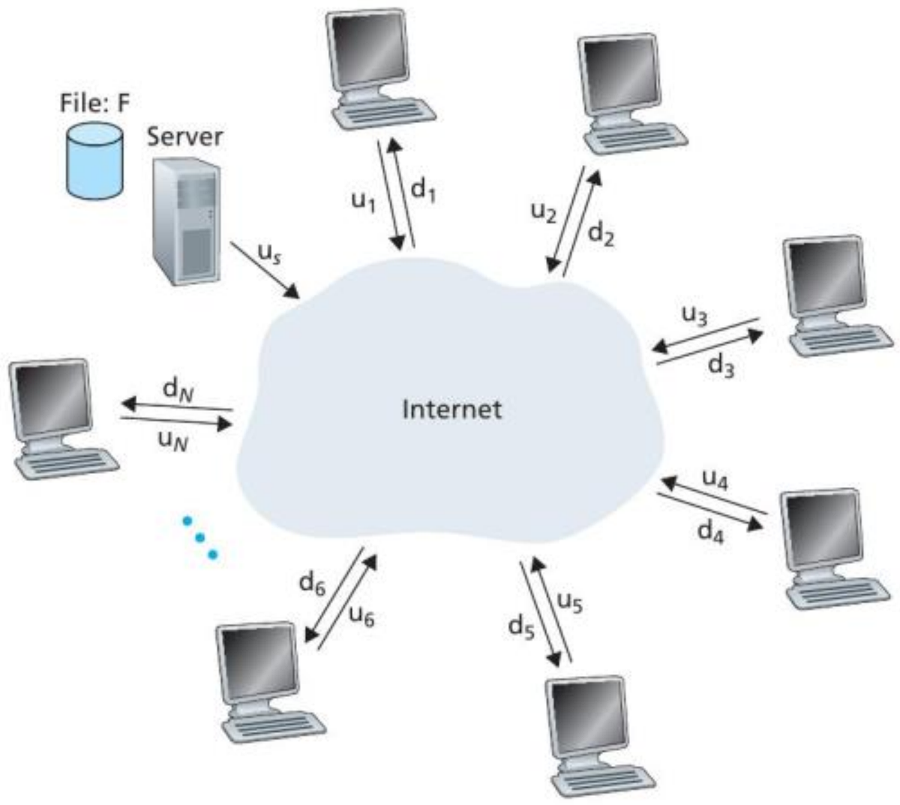
Applications of Peer-to-Peer Architecture

- File distribution: application that transfers a file from a single source to multiple peers.
- Database distributed over a large community of peers.
- Internet telephony : Skype.

File Distribution:

- Each peer can redistribute any portion of the file to any other peer
- Popular file distribution protocol : BitTorrent,developed by Bram Cohen
- Scalability

Scalability



- N peers
- Distribution time: the time required to distribute a file to all peers.

Assumptions

- Internet has abundant bandwidth and all bottlenecks are in the network access
- All the server and client bandwidth is available for file distribution

Distribution Time for Client-Server Architecture

- Let D_{cs} denote the distribution time for client-server architecture for a file size of F bits
- The server has to transmit a total of NF bits at an upload rate of $u_s bps$.
- Minimum time required for distribution is $\frac{NF}{u_s}$ seconds
- Let $d_{min} = \min\{d_1, \ldots, d_N\}$
- Minimum distribution time is $\frac{F}{d_{min}}$ seconds
- Thus,

$$D_{cs} \geq \max\left\{\frac{NF}{u_s}, \frac{F}{d_{min}}\right\}$$

- Show that

$$D_{cs} = \max\left\{\frac{NF}{u_s}, \frac{F}{d_{min}}\right\}$$

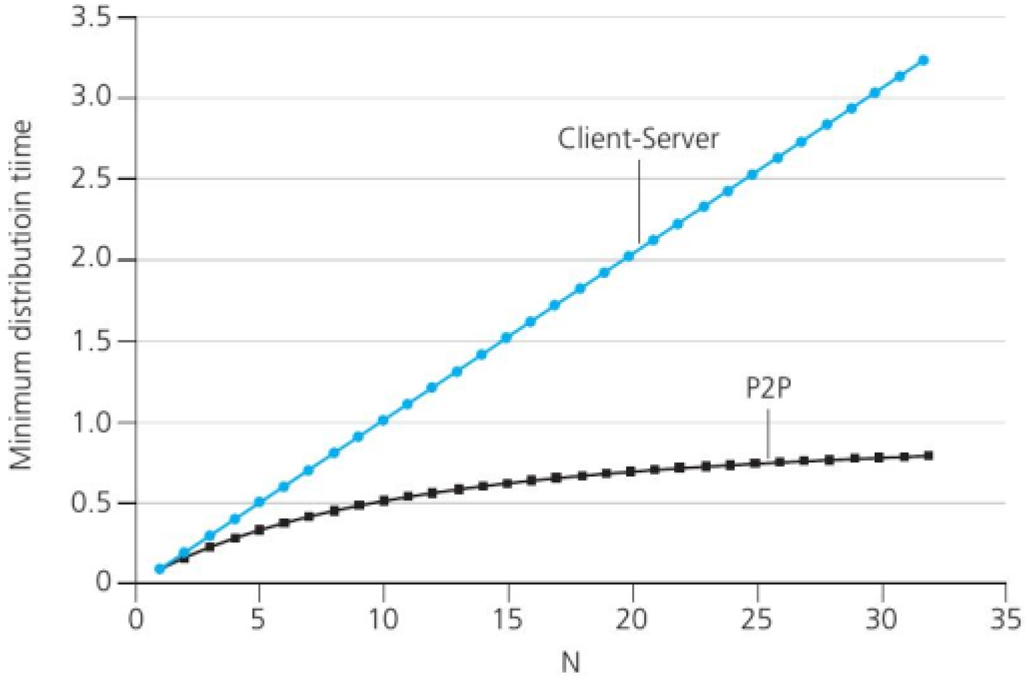
Distribution Time for P2P Architecture

- The server has to send each bit of the file at least once: Minimum distribution time is at least $\frac{F}{u_s}$ seconds
- The peer with lowest download rate can not obtain F bits in less than $\frac{F}{d_{min}}$ seconds
- The total upload rate $u_{total} = u_s + u_1 + \cdots + u_N$. The system must deliver F bits to each of the N peers: Minimum distribution time is $\frac{NF}{u_{total}}$
- Thus, minimum distribution time D_{P2P} is at least

$$\max\{\frac{F}{u_s}, \frac{F}{d_{min}}, \frac{NF}{u_{total}}\}$$

- Assumption: each peer can redistribute a bit as soon as it receives the bit.
- There is a scheme that actually achieves this lower bound.

Distribution Time for P2P Architecture



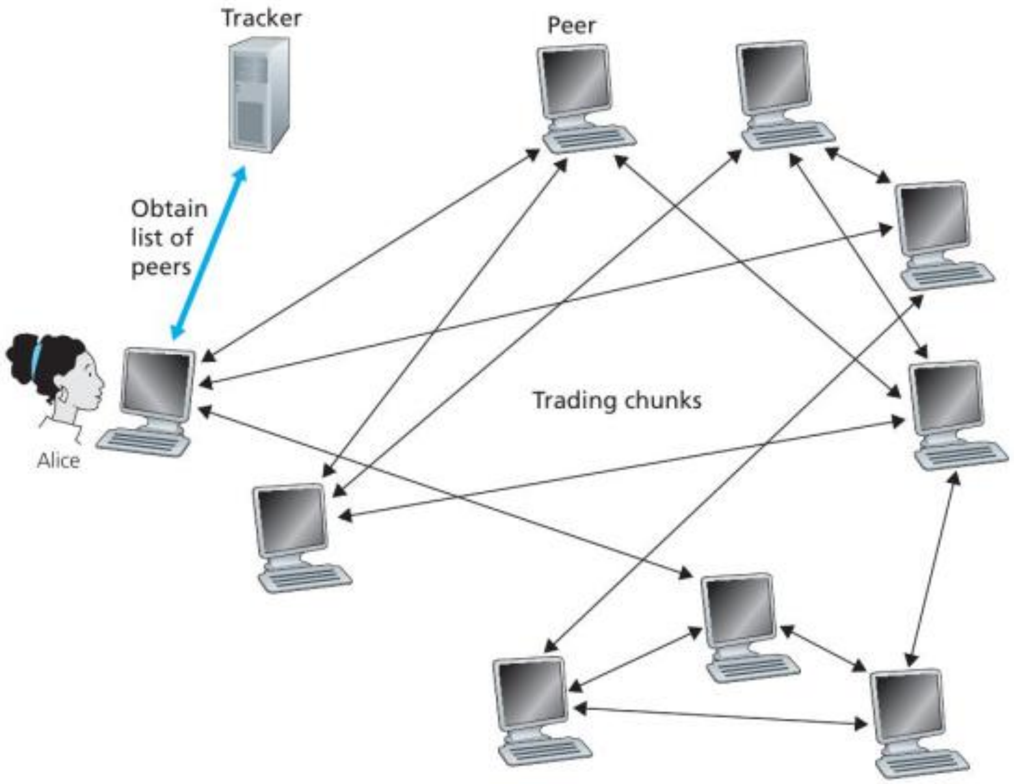
- All peers upload at a rate of u bps.
- $\frac{F}{u} = 1$ hour, $u_s = 10u$ and $d_{min} \geq u_s$.

Bit Torrent

- Collection of peers participating in the distribution of a file is called a torrent
- Peers in a torrent download equal-size chunks of the file (typically 256 KBytes)
- A peer accumulates more and more chunks over time
- Once a peer has acquired complete file, it may leave the torrent or continue to participate in the torrent
- Peers may leave torrents with subsets of chunks

Bit Torrent

- Each torrent has a node called tracker.
- When a peer joins the torrent, it registers with the tracker
- Each peer in the torrent periodically updates the tracker about its presence.



Bit Torrent

- Alice receives a subset of participating peers in the torrent
- She establishes TCP connection with some of the peers and we call them as neighboring peers of Alice
- Neighboring peers may vary over time
- Each peer will have some subset of chunks from the file, with different peers having different subsets
- Alice maintains a list of chunks that her neighbors have.

Bit Torrent

- Alice will issue requests for chunks she currently does not have
- Which chunks should be requested first?
- Rarest first: finds the chunks that are rarest among her neighbors

- Alice will issue requests for chunks she currently does not have
- To which of her neighbors should she send requested chunks?
- Tit-for-tat

Tit-for-tat

- Alice gives priority to the neighbors that are currently supplying her data at the highest rate
- Typically four neighbors are chosen. These peers are said to be unchoked
- Every 30 seconds, she also picks one additional neighbor at random and sends it chunks. Let it be Bob.
- Bob is said to be optimistically unchoked.
- In due course of time, Alice, may become one of the top uploaders in which case Bob could start sending data to Alice.

Distributed Hash Tables (DHT)

- Huge database to be stored among number of peers in a distributed way
- Database is consists of (key, value) pairs. For Example, (PAN No., Aadhar No.), (Content Name, IP), etc.
- Peers query the database by supplying the key and database replies the matching pairs to the querying peer
- How to store database among the peers

DHT

- Assign an identifier to each peer.
- An identifier is an integer in $[0, 2^n - 1]$ for some fixed n
- (key, value) pairs are also identified by integers using hash functions
- Hash function is available to all peers.

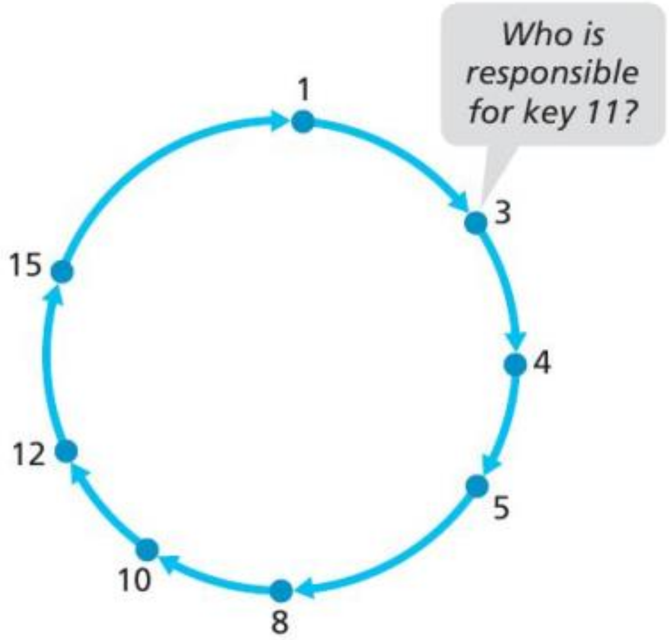
problem of designing a DHT for general key- value pairs.

One naive approach: to randomly scatter the (key, value) pairs across all the peers
→each peer maintain a list of the IP addresses of all participating peers.
→ the querying peer sends query to all other peers → peers containing the (key, value) pairs that match the key can respond with their matching pairs → completely unscalable
→ require each peer to not only know about all other millions of peers → worse, have each query sent to all peers.

Storing in DHT

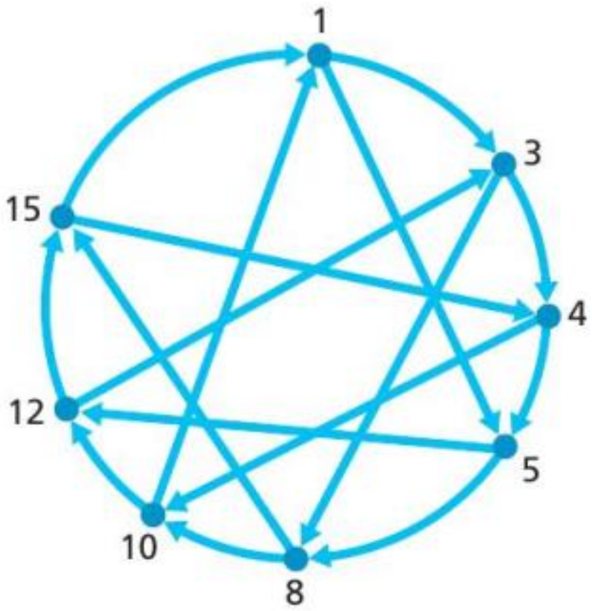
- Define a rule for assigning keys to peers
- Closest to the key:
- For example, $n = 4$, with eight peers: 1,3,4,5,8,10,12 and 15. Store (11, 0123-4567-8910) in one of the eight peers
- By closest convention, peer 12 is the immediate successor for key 11. Store in peer 12.
- If the key is larger than all the peer identifiers, we use modulo-2^n convention.

Circular DHT



- Each peer is aware of only its immediate predecessor and successor
- N messages at most

Shortcut



- Number of shortcuts are relatively small in number
- How many shortcut neighbors and which peers should be these shortcut neighbors? Research problem: $O(\log(N))$

Peer Churn

- Peers can come and go without warning
- Peers keep track to two immediate predecessor and successors.
- When a peer abruptly leaves, its predecessor and successor learn that a peer has left and updates the list of its predecessor and successor.