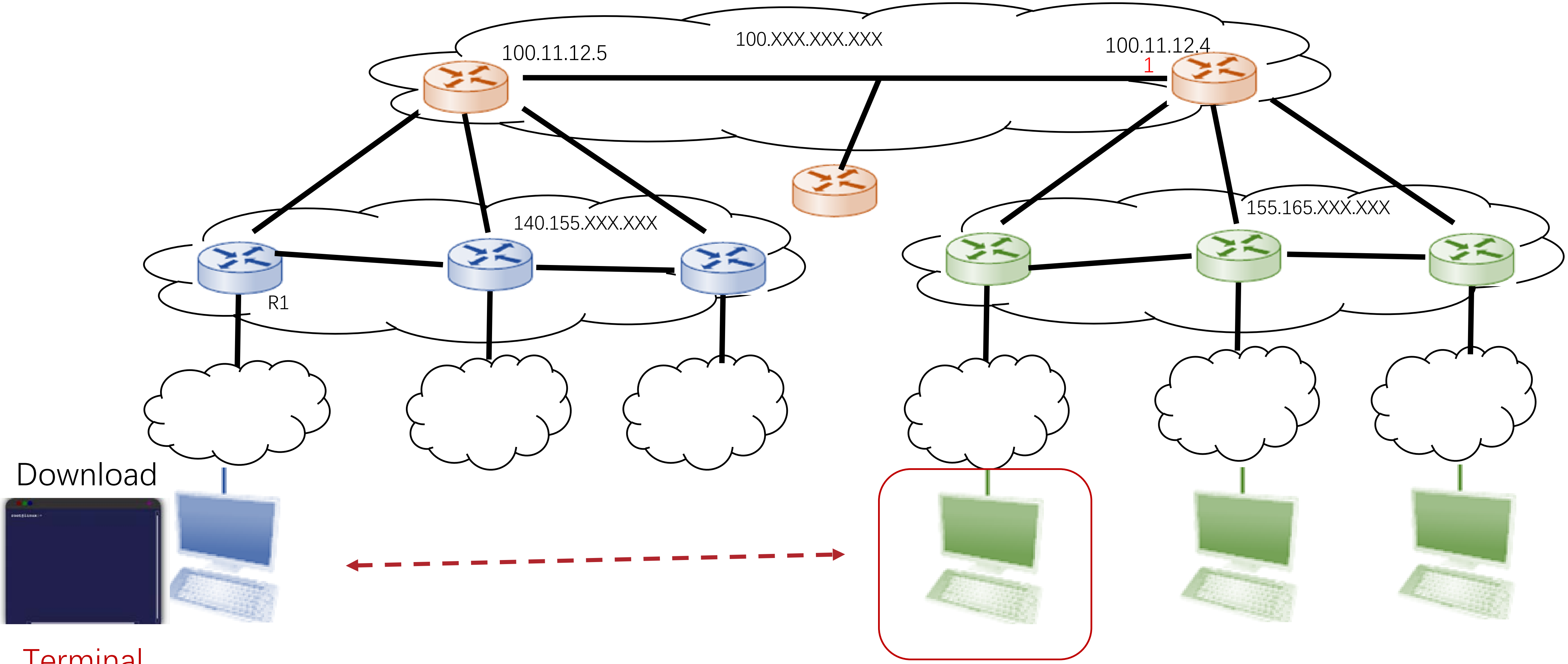
CS120: Computer Networks

Lecture 26. FTP & P2P

Zhice Yang

File Service



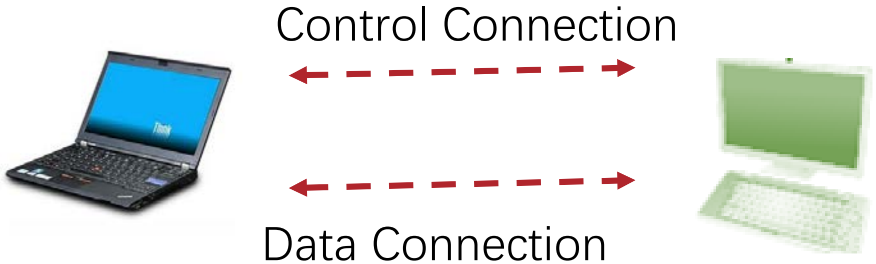
Download

Terminal

myserver.sist.shanghaitech.edu.cn

File Transfer Protocol

- FTP: RFC 959
- Use TCP
- Two Connections
  - Control Connection
    - Server Port 21
    - Control Command
    - Authentication
    - Show Directory
  - Data Connection
    - Open one TCP connection for transferring a data stream
    - One data stream one data connection
- Two Working Mode
  - Passive Mode: client connects to server for data connection
  - Active Mode: server connects to client for data connection



File Transfer Protocol

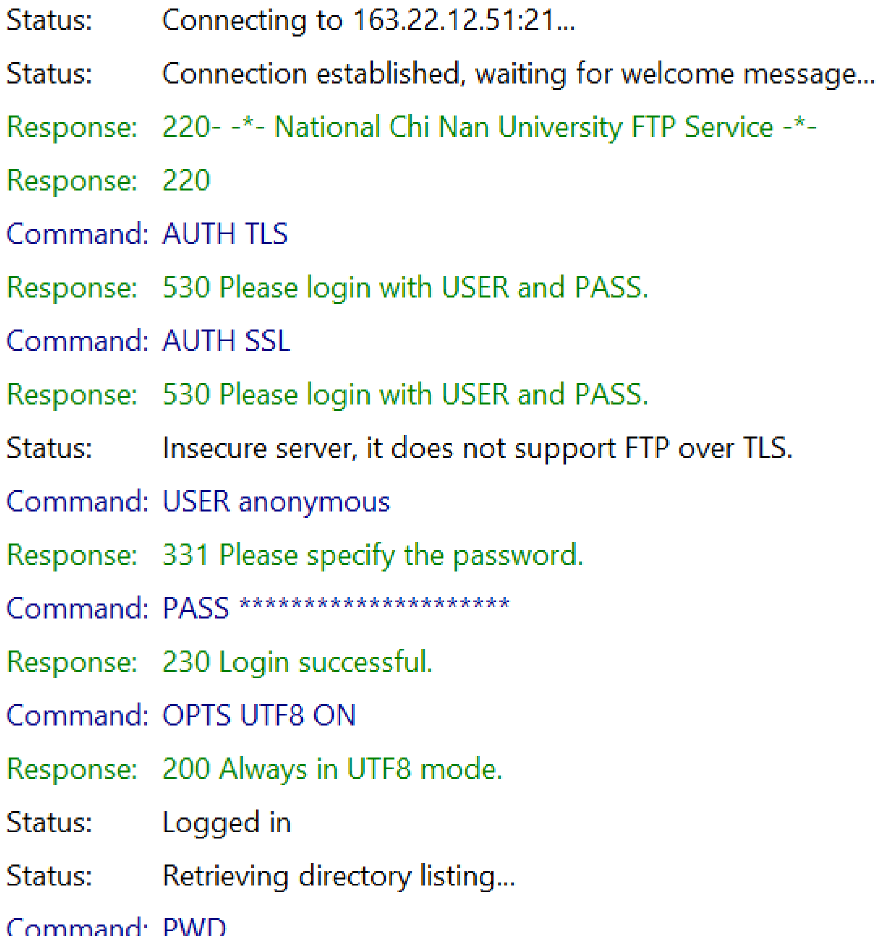
- Control Connection
  - Like HTTP, Messages are Text-oriented

ABOR - **abor**t a file transfer
CWD - **c**hange **w**orking **d**irectory
DELE - **dele**te a remote file
LIST - **list** remote files
MDTM - return the **m**o**d**ification **t**i**m**e of a file
MKD - **m**a**k**e a remote **d**irectory
NLST - **n**ame **l**i**st** of remote directory
PASS - send **pass**word
PASV - enter **pas**si**v**e mode
PORT - open a data **port**
PWD - **p**rint **w**orking **d**irectory

QUIT - terminate the connection
RETR - **retr**ieve a remote file
RMD - **r**e**m**ove a remote **d**irectory
RNFR - **r**e**n**ame **fr**om
RNTO - **r**e**n**ame **to**
SITE - **site**-specific commands
SIZE - return the **size** of a file
STOR - **stor**e a file on the remote host
TYPE - set transfer **type**
USER - send **user**name

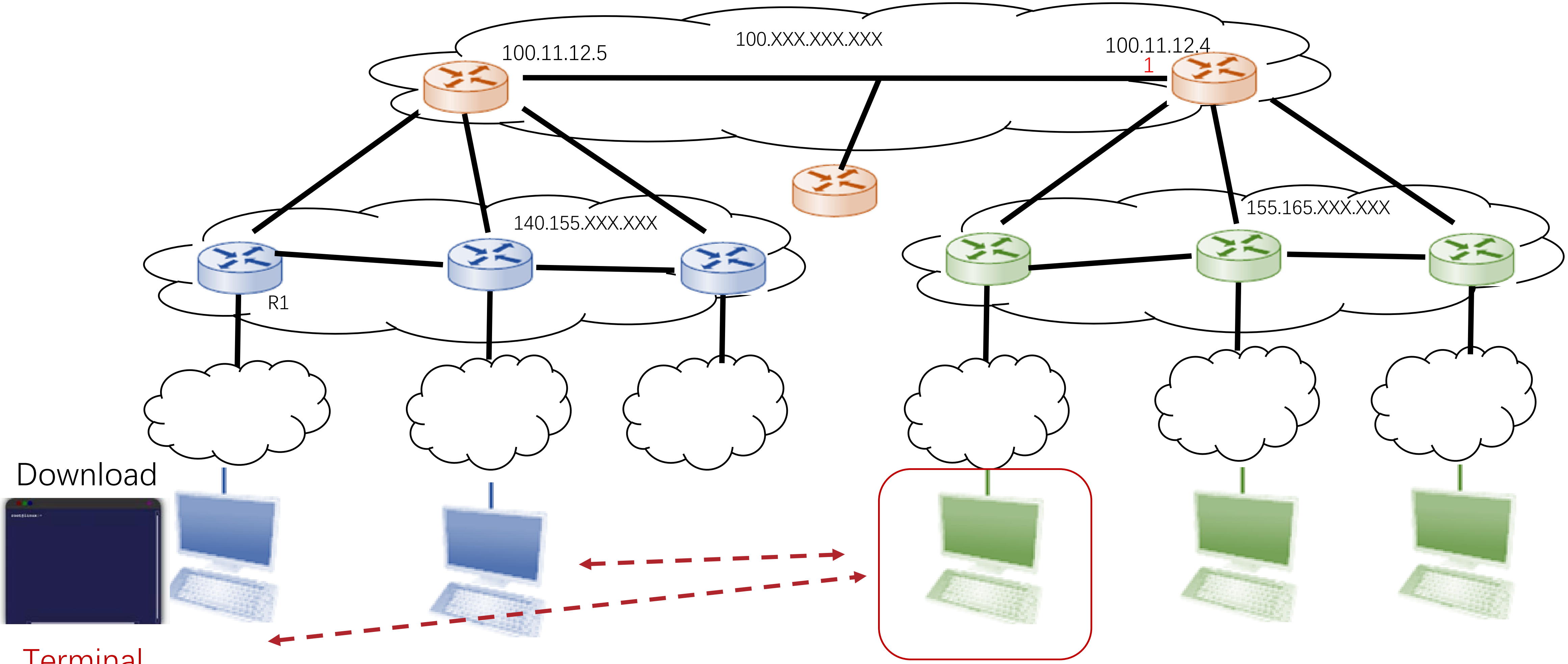
Demo

- Telnet
- FileZilla



```
Status:     Connecting to 163.22.12.51:21...
Status:     Connection established, waiting for welcome message...
Response:   220- -*- National Chi Nan University FTP Service -*-
Response:   220
Command:  AUTH TLS
Response:   530 Please login with USER and PASS.
Command:  AUTH SSL
Response:   530 Please login with USER and PASS.
Status:     Insecure server, it does not support FTP over TLS.
Command:  USER anonymous
Response:   331 Please specify the password.
Command:  PASS *********************
Response:   230 Login successful.
Command:  OPTS UTF8 ON
Response:   200 Always in UTF8 mode.
Status:     Logged in
Status:     Retrieving directory listing...
Command:  PWD
```

File Service for Multiple Clients

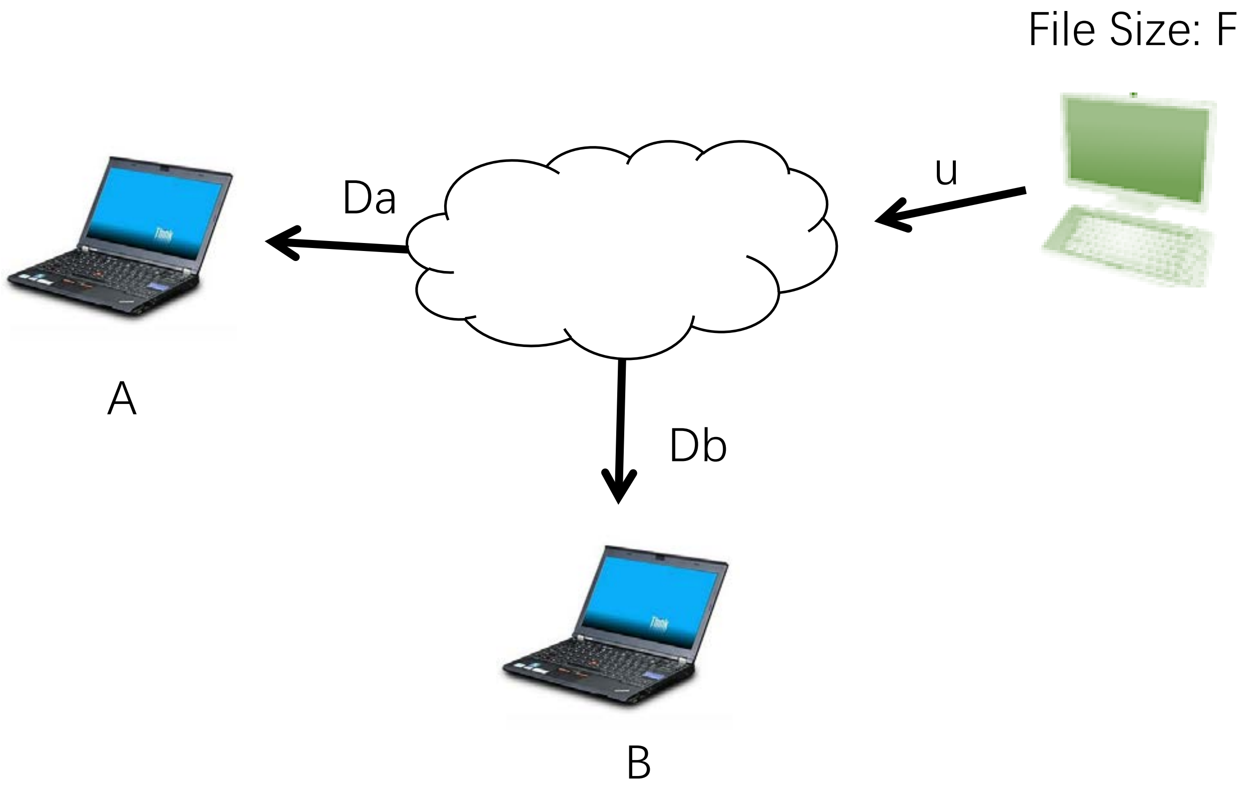


```
root@linux:~
```

myserver.sist.shanghaitech.edu.cn

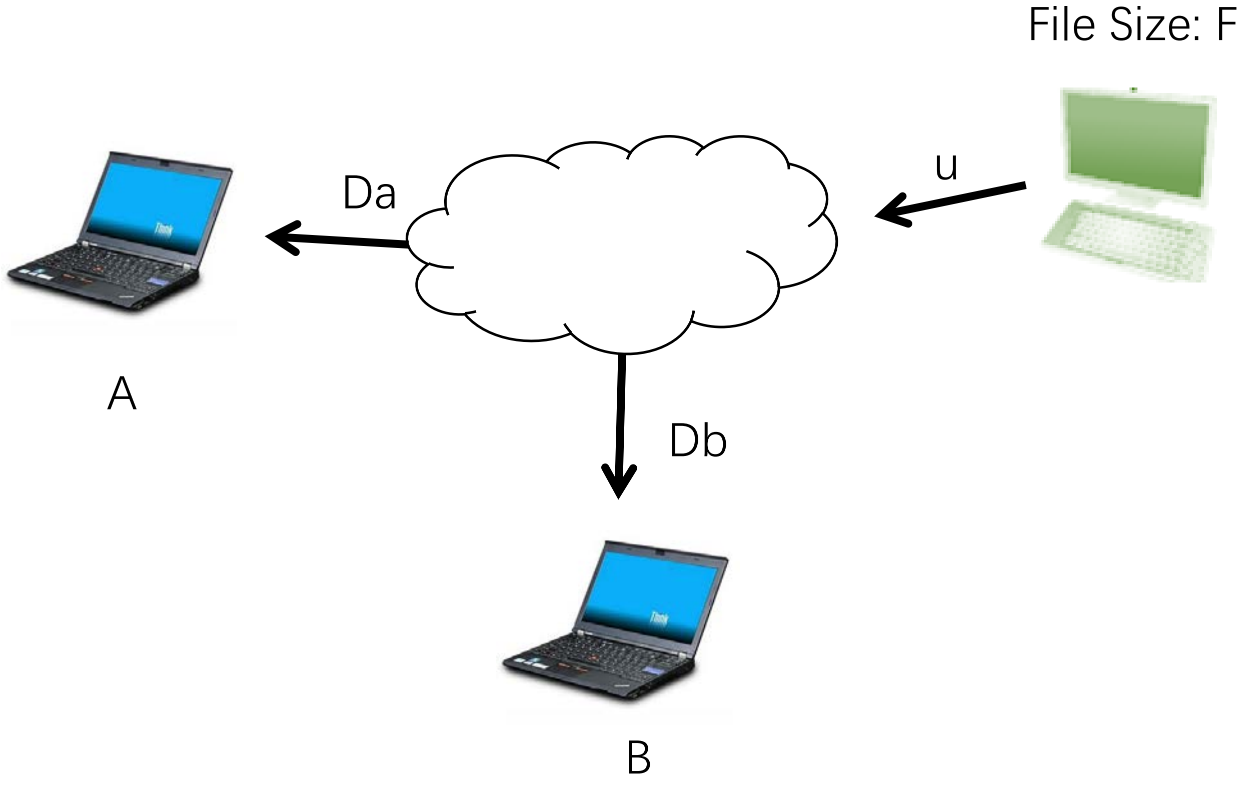
File Service for Multiple Clients

- Simple Approach
  - Server: sequentially send (upload) file copies
  - Client: download file copy
  - Total Time
    - Max {2∗F/u, F/Da, F/Db}



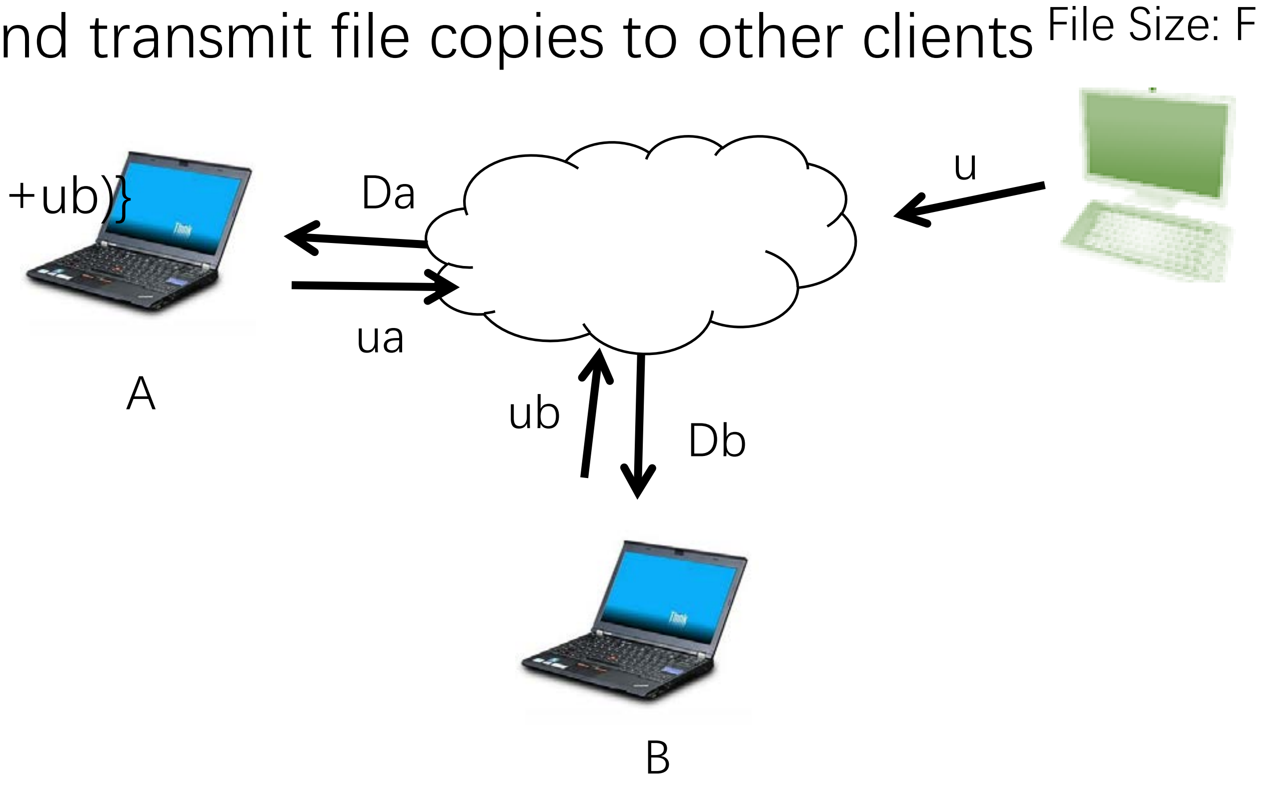
File Service for Multiple Clients

- Multicast Approach
  - Server: broadcast (upload) file copies to clients
  - Client: download file copy
  - Total Time
    - Max {F/u, F/Da, F/Db}



File Service for Multiple Clients

- Peer to Peer (P2P) Approach
    - Server: transmit (upload) file copies to clients
    - Client: download file copies and transmit file copies to other clients
    - Total Time
        - Max {F/u, F/Da, F/Db, 2F/(u+ua+ub)}

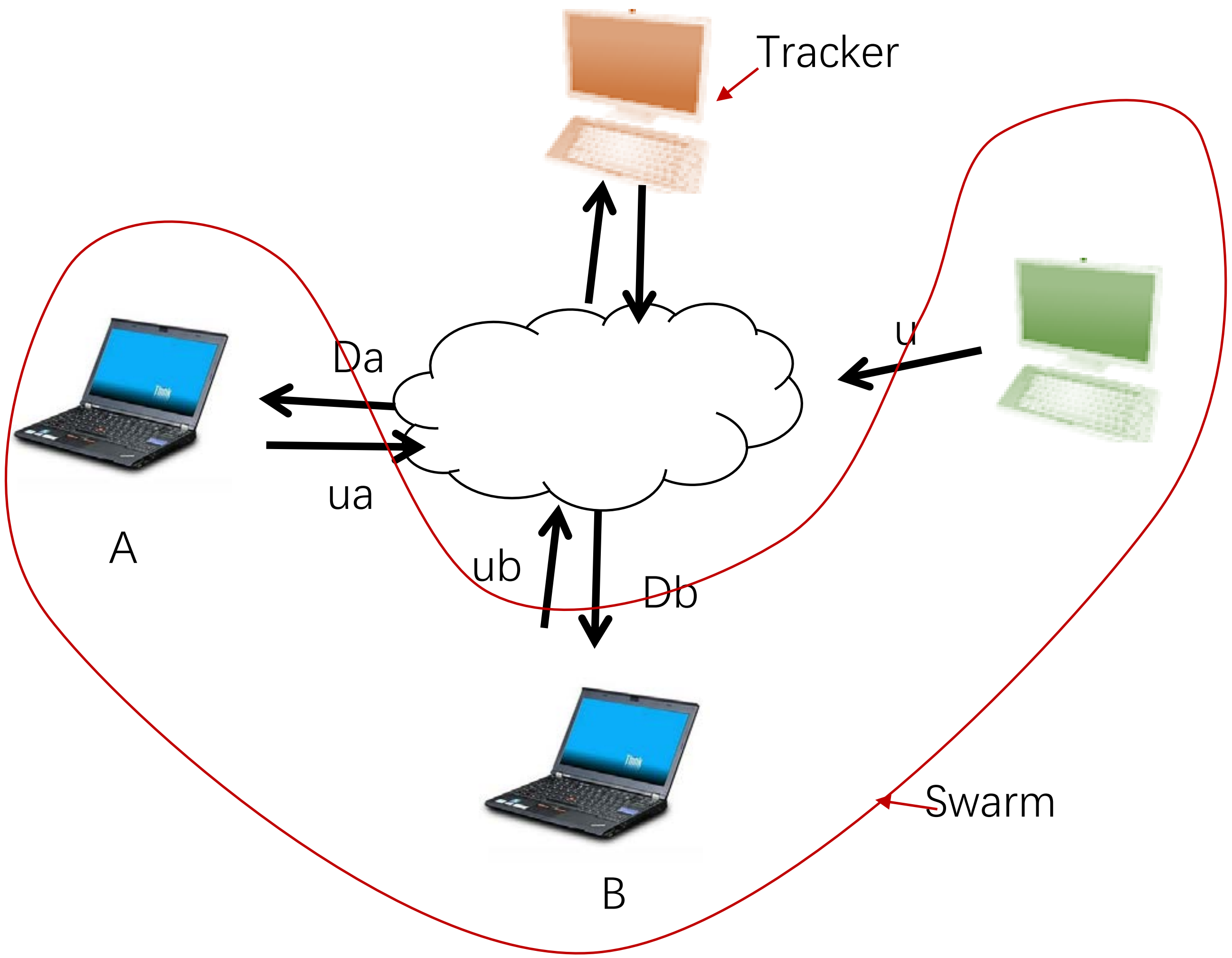


P2P File Distribution: BitTorrent

- BitTorrent is a P2P file sharing system
  - Client: BitTorrent, uTorrent, Thunder, etc

BitTorrent

- The tracker is a central server keeping a list of all peers participating in the swarm
- A swarm is the set of peers that are participating in distributing the same files
- Peer joins a swarm by asking the tracker for a peer list and connects to those peers

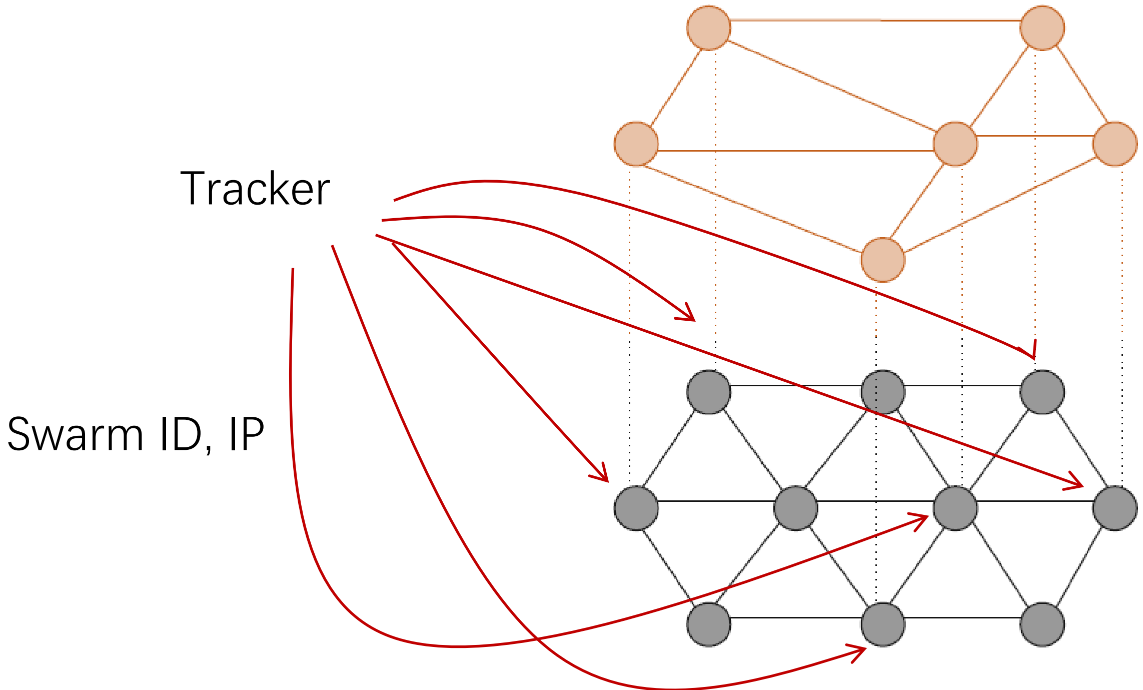


BitTorrent

- A metadata file (.`torrent`) is distributed to all peers
  - Usually via HTTP
  - .`torrent` is encoded with “bencode”
    - online tools to decode: https://www.tools4noobs.com/online_tools/torrent_decode/
  - The metadata contains
    - File names
    - SHA-1 hashes of all pieces of the file
      - http://www.sha1-online.com/
    - Tracker’s url
    - Tracker list
    - Info-hash
    - etc.

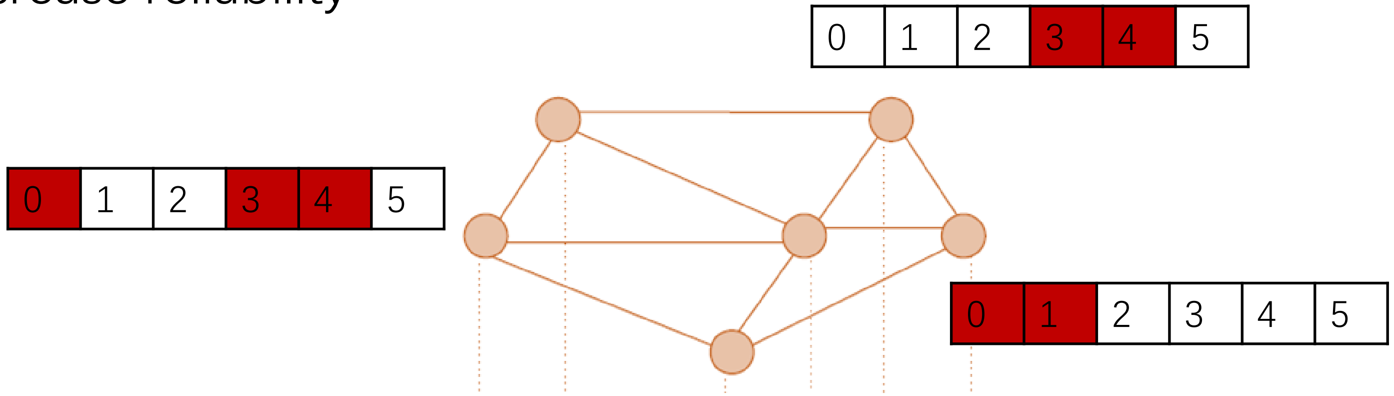
BitTorrent

- The Overlay Networks in P2P
    - Tracker tracks peer information
    - New peer registers with tracker to get list of peers
    - Download files from peers through TCP



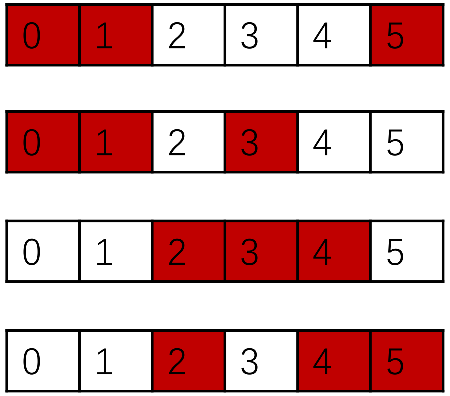
BitTorrent

- File Distribution
  - Peers may have different pieces of file
    - Upload pieces while downloading
  - New peer has no pieces
    - But will accumulate over time
  - Peers exchange information of the pieces they have
    - To maximize throughput
    - To Increase reliability

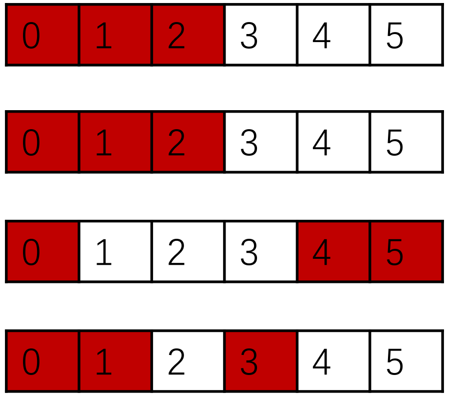


BitTorrent

- Piece Overlap
  - Big overlap -> Only a few peers can exchange pieces
  - Minimize piece overlap
    - Download random pieces
    - Priorities the rarest pieces, aiming towards uniform piece distribution



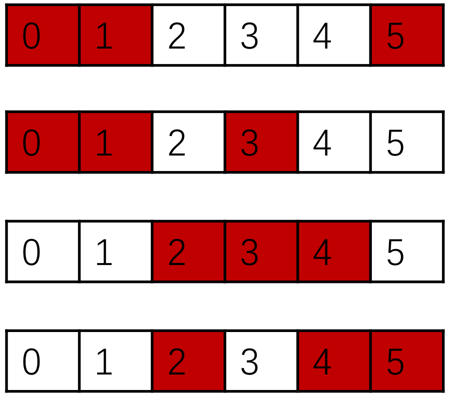
Small overlap



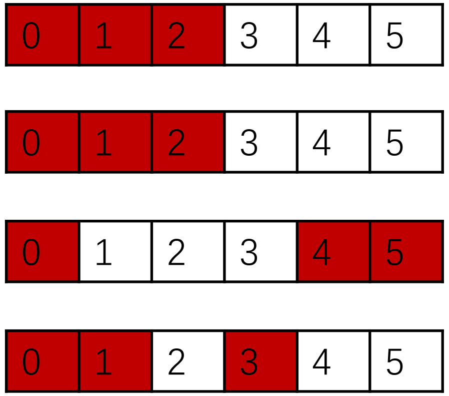
Big overlap

BitTorrent

- Piece Redundancy
  - Be tolerant against dropping peers
  - Maximize piece redundancy
    - Maximize the number of distributed copies (the rarest pieces)
      - Download the rarest pieces first



Distributed copies = 2



Distributed copies = 1

BitTorrent

- The Last Piece
  - The download time of the last piece could be longer than other pieces
    - Pieces with fast download speed have been finished
  - Increase download choices for the last piece
    - Assign more peers to transmit

BitTorrent

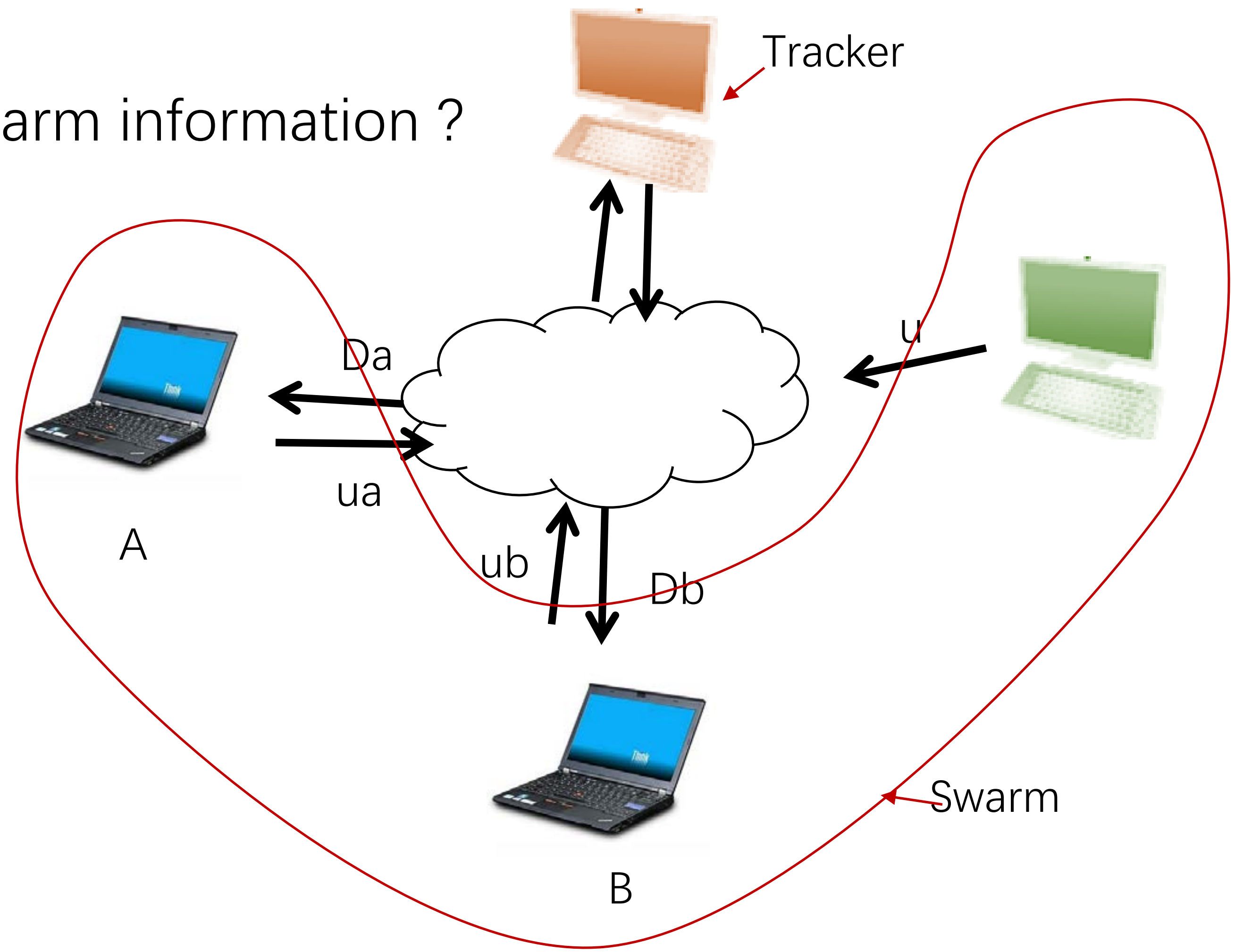
- The Piece Picking Policies
  - Random First Piece
  - Rarest Piece First
  - The End Game Mode
    - Send request to all peers to download the last piece

BitTorrent

- The Incentive to Share
  - There is a loose connection between upload and download speed
  - Each peer has an incentive to upload

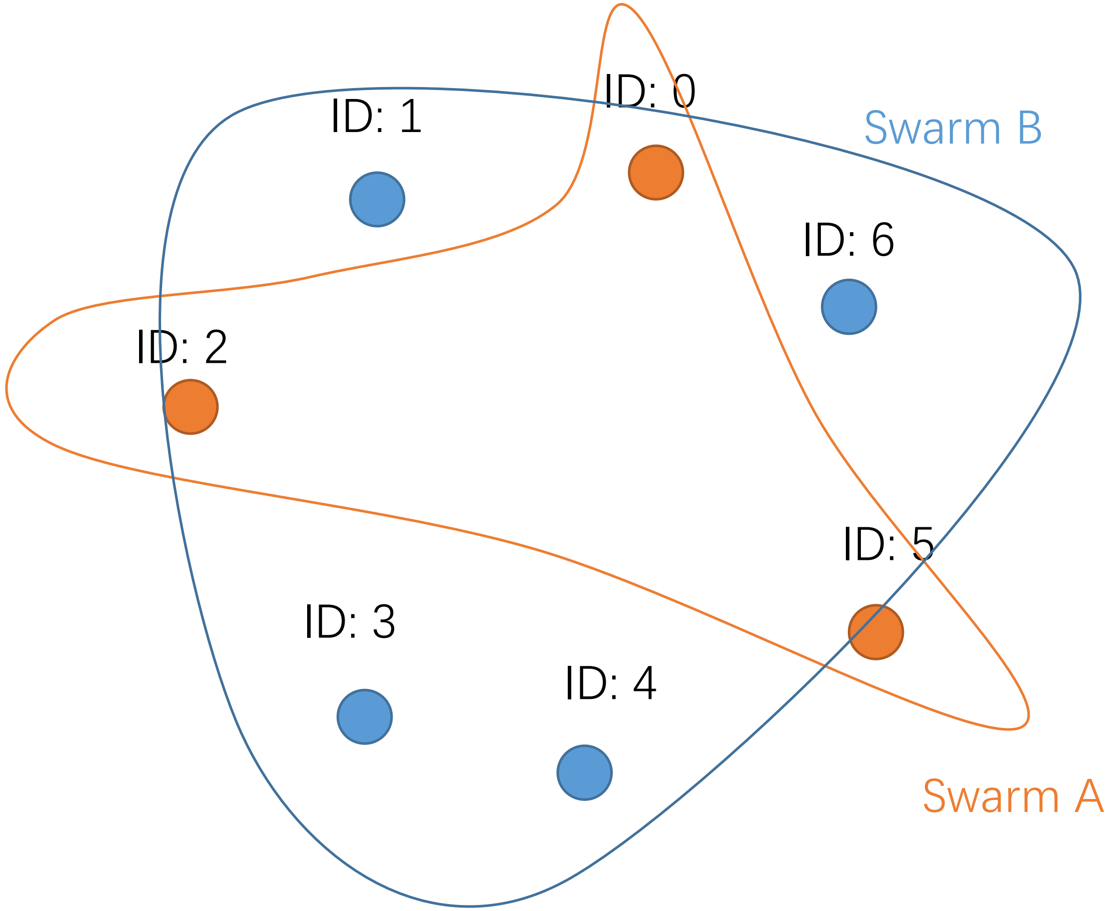
BitTorrent

- Trackerless Design
    - Where to store the swarm information ?



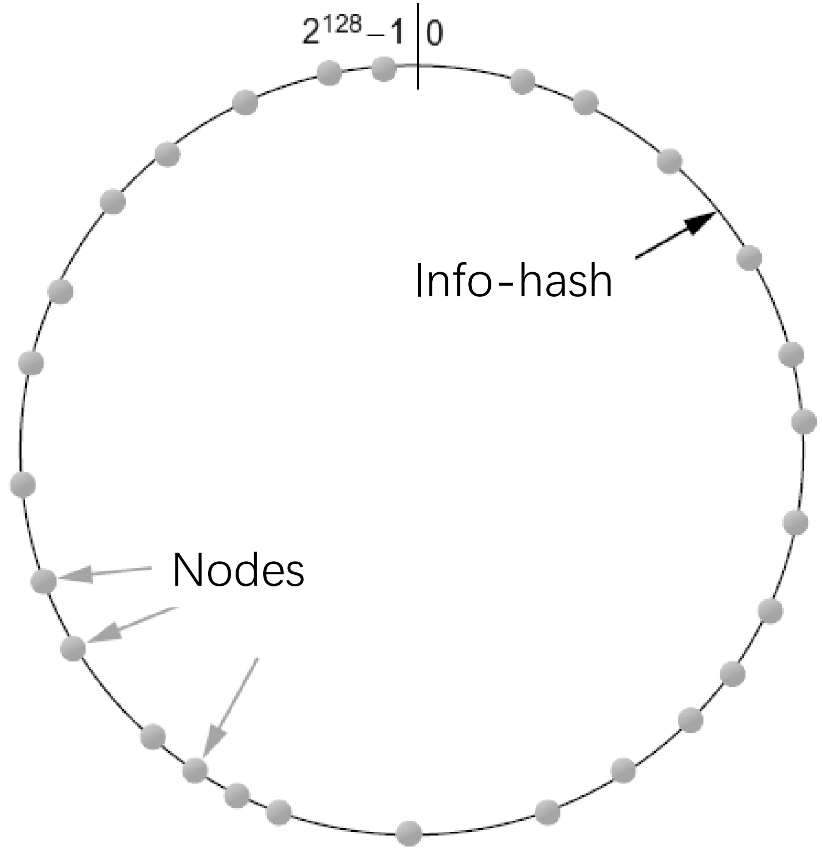
Distributed Hash Table

- Distributed Hash Table (DHT)
  - Hash Table: <key, value>
    - Hash(key) -> value
  - BitTorrent DHT:
    - "key" is the info-hash, ie. the hash of the metadata of the torrent file.
    - "value" is the peer list of the swarm
    - Hash(info-hash) -> peers info



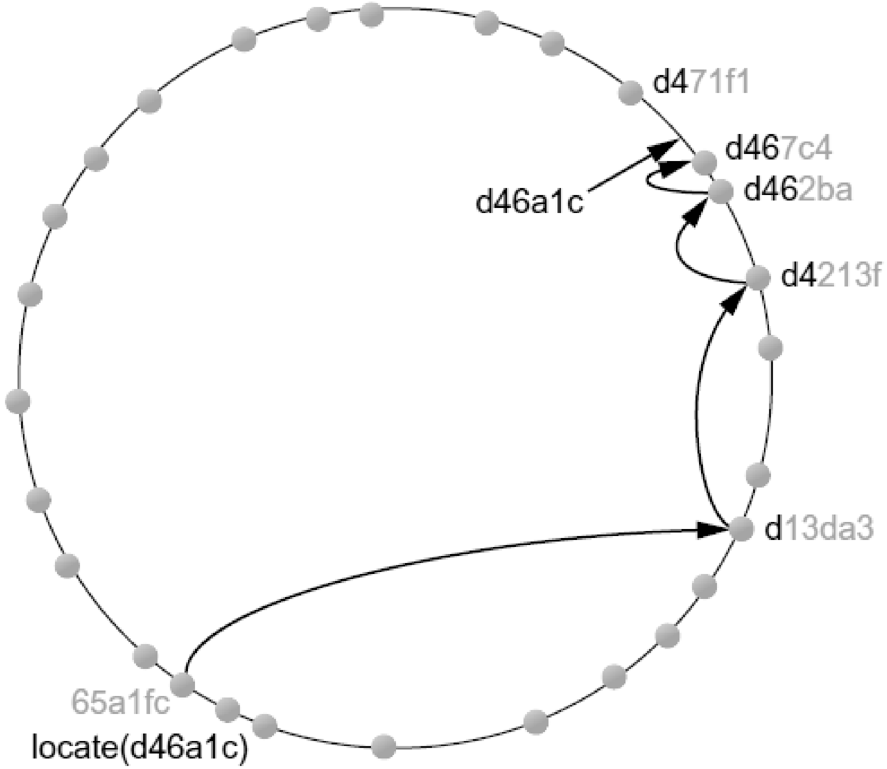
Distributed Hash Table

- Basic Idea:
  - Key (info-hash) is an integer
  - Assign an integer ID to each node
  - Map key and node ID into the same space
- Key: Info-hash from `.torrent`
- Each node randomly choose an ID
  - Same as the key space (160 bits)
- Store the peer list of a torrent in the node whose ID is closest to the info-hash of the torrent



Distributed Hash Table

- Find the Peer List
  - Obtain the Info-hash (e.g. d46a1c)
  - Route to the closest node to d46a1c
    - Each node has a partial routing table (not a complete one)
      - Initial routing table is obtained from torrent file or previous known nodes
    - The routing table contains IPs of certain IDs
    - Iteratively forward the route query to the node with closer ID
      - According the prefix of the IDs
    - Node having the closest ID replies the peer list
      - e.g. d467c4 node replies
  - Add itself to the peer list
    - e.g. add 65a1fc to d467c4



Distributed Hash Table

- Add to the Peer List (Practical Way)
  - Each peer announces itself with the distributed tracker
    - Looking up the 8 nodes closest to the info-hash of the torrent
      - The 8 nodes’ IPs are stored in the torrent file
    - Send an announce message to them
    - Those 8 nodes will then add the announcing peer to the peer list stored at that info-hash
    - Each announce looks up new nodes, in case nodes have joined the network with IDs closer to the info-hash than a previous node

Reference

- Textbook 9.4
- http://www.bittorrent.org/beps/bep_0005.html
- https://www.youtube.com/watch?v=YFV908uoLPY