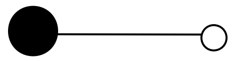
# CSc 461/561
# Multimedia Systems
# Peer-to-Peer Multimedia Delivery

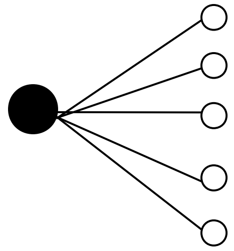
Jianping Pan

Spring 2015

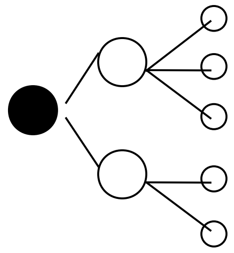
# Service models



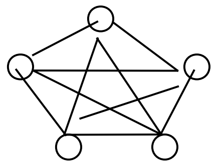
- Client-Server
  - request-reply transaction
  - scalability concern



- Client-Intermediary-Server
  - intermediary: e.g., proxy server
  - content distribution server



- Peer-to-Peer
  - client/server-server/client



# Peer-to-Peer model

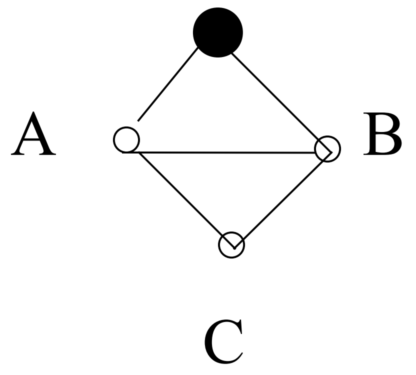
- "Serving while being served"
  - retrieve data from somewhere (other peers)
  - use the data
  - and then
    - store the data, throw the data away, or
    - *recycle* the data for other peers
- Examples: peer-to-peer file sharing
  - (old) Napster, Gnutella, BitTorrent, etc

# Peer-to-Peer challenges

- Service discovery
  - no longer always from a well-known server
  - central directory, flooding query, DHT, etc
- Information delivery
  - from (many) other peers
  - peer collaboration
- Clients are *not* servers
  - how to handle client join/leave

## Peer-to-Peer content distribution

- Example
  - content server: 1, 2, 3, 4, 5
  - client A gets: 1, 3, 5
  - client B gets: 2, 4
  - client C gets: 1, 3, 5 from A and 2, 4 from B



- Compare
  - client A/B/C gets 1,2,3,4,5 from the server
- Explore *network* diversity

## Service discovery (1)

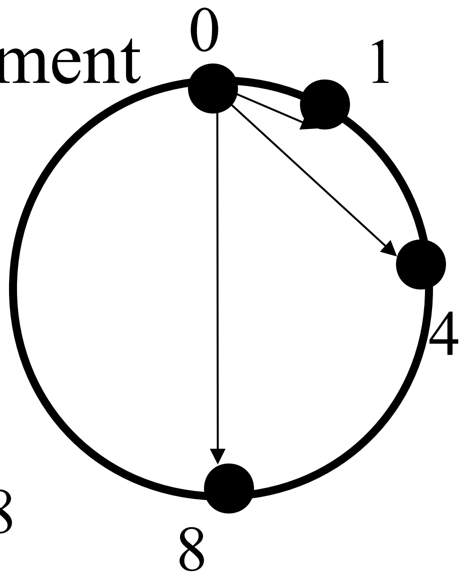
- Directory server
  - e.g., Napster
  - register resources with directory server(s)
  - inquire directory server for resource location
  - retrieve resources from peers directly
    - resource: peer-to-peer
    - resource meta-info: client-server
  - bottleneck at directory server

## Service discovery (2)

- Flood query
  - e.g., Gnutella
  - flood neighbor with queries
    - recursive until TTL expires
  - “slow” search
- Hybrid mode
  - peer and super-peer
  - super-peers keep resource information for peers

## Service discovery (3)

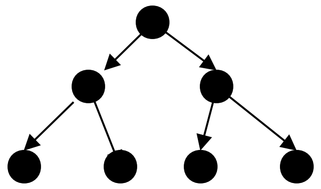
- Distributed hash tables
  - e.g., Chord, CAN, etc
  - hash resource, peer ID into a key space
    - e.g., a circular space
  - peers are responsible for a space segment
    - put(key); get(key)
  - neighbor and finger routing
- Structured vs unstructured



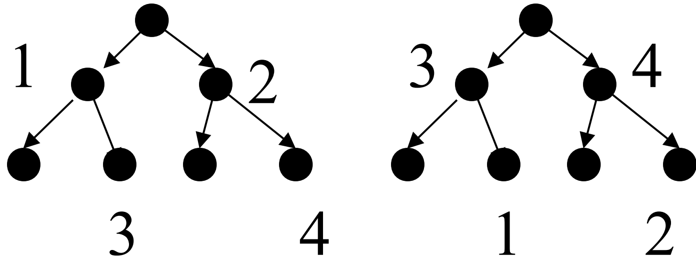
# Peer-to-Peer streaming

- More challenges
  - data order matters for streaming applications
    - e.g., in the last example
      - until data block 1 comes, client 2 cannot start playback
  - much more peer dynamics during streaming
    - longer session time
    - peers can join/leave at anytime
      - find better peers
      - replace bad peers
      - minimize impact due to departing peers

# Building trees



- One tree
  - internal nodes vs leaf nodes
- Multiple trees
  - split information into parts
    - FEC, layers, multiple descriptors
  - send one part along one tree
  - peers reconstruct information with some parts
  - peers: internal node in one tree, leaf nodes in others



# Peer collaboration

- Reputation-based mechanisms
  - e.g., closed BitTorrent communities
  - “second-hand” information
- Payment-based mechanisms
  - micro-payment
- Score-based mechanisms
  - send/receive >= 1
  - heterogeneous peers

# This lecture

- Peer-to-peer multimedia delivery
  - service models
  - peer-to-peer structures
  - service discovery
  - peer-to-peer streaming