Sistemas Distribuídos
Aula 9

Roteiro

- Arquitetura P2P
- Bittorrent
- Distributed Hash Table (DHT)

Aquitetura de Sistemas Distribuído

- Duas grandes abordagens
  - cliente/servidor: clássica e mais usada
  - P2P (peer-to-peer): mais nova e mais sofisticada
- Cada qual com vantagens e desvantagens
- Não necessariamente ortogonais
  - sistemas reais misturam as duas abordagens

Definições não são cartesianas!

Peer-to-Peer (P2P)

- Componentes separados em máquinas que fazem papéis semelhantes
- **Ideia central**: *cliente também é servidor*
  - demanda gerada por clientes pode ser atendida por outros clientes
- Recursos disponíveis nos clientes usados para prover serviço para outros clientes
  - CPU, disco, banda, etc

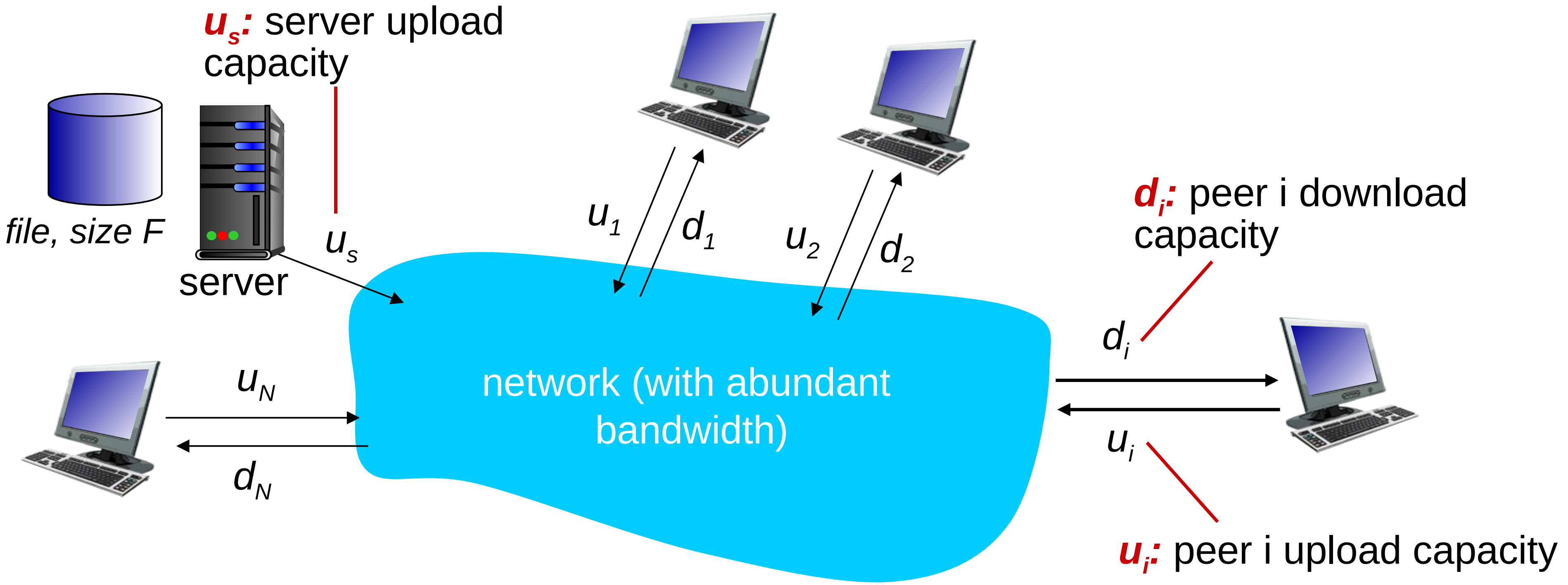
Maior Escalabilidade!

- Fundamentalmente diferente de cliente/servidor
- Muito mais “sistema distribuído”, mais difícil
  - ex. intermitência (entra e sai) dos pares

Distribuir Arquivo: C/S versus P2P

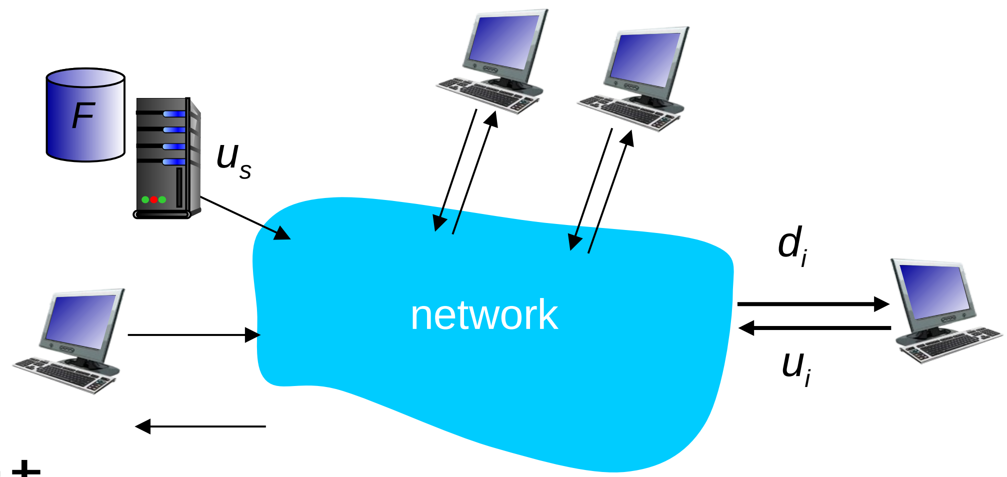
Quanto tempo é necessário para distribuir arquivo (*F* bytes) de um servidor para *N* computadores?

- capacidade de upload/download é recurso limitado



File distribution time: client-server

- *server transmission:* must sequentially send (upload) *N* file copies:
  - time to send one copy: F/u_s
  - time to send N copies: NF/u_s



- *client:* each client must download file copy
  - d_{min} = min client download rate
  - min client download time: F/d_{min}

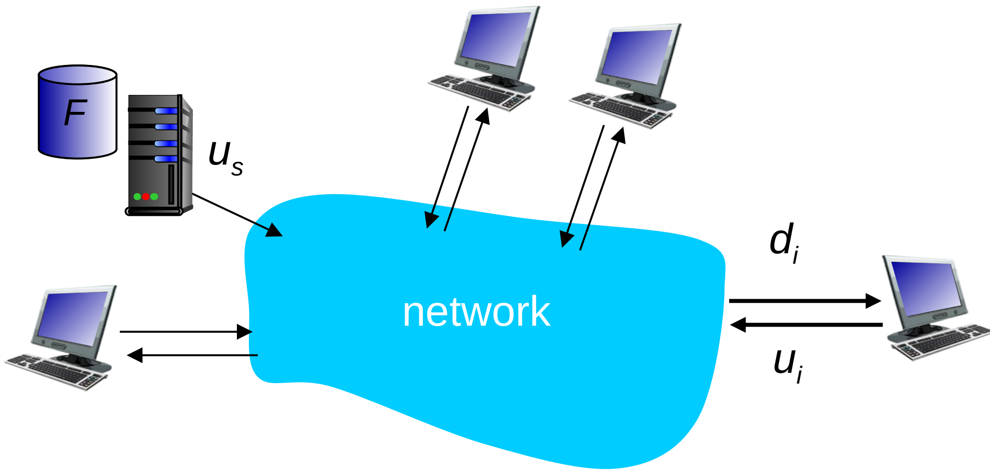
time to distribute F to N clients using client-server approach

$$D_{c\text{-}s} > max\{NF/u_s, F/d_{min}\}$$

increases linearly in N

File distribution time: P2P

- *server transmission:* must upload at least one copy
  - time to send one copy: F/u_s
- *client:* each client must download file copy
  - min client download time: F/d_{min}
- *clients:* as aggregate must download *NF* bits
  - max upload rate (limiting max download rate) is $u_s + \Sigma u_i$



time to distribute F to N clients using P2P approach

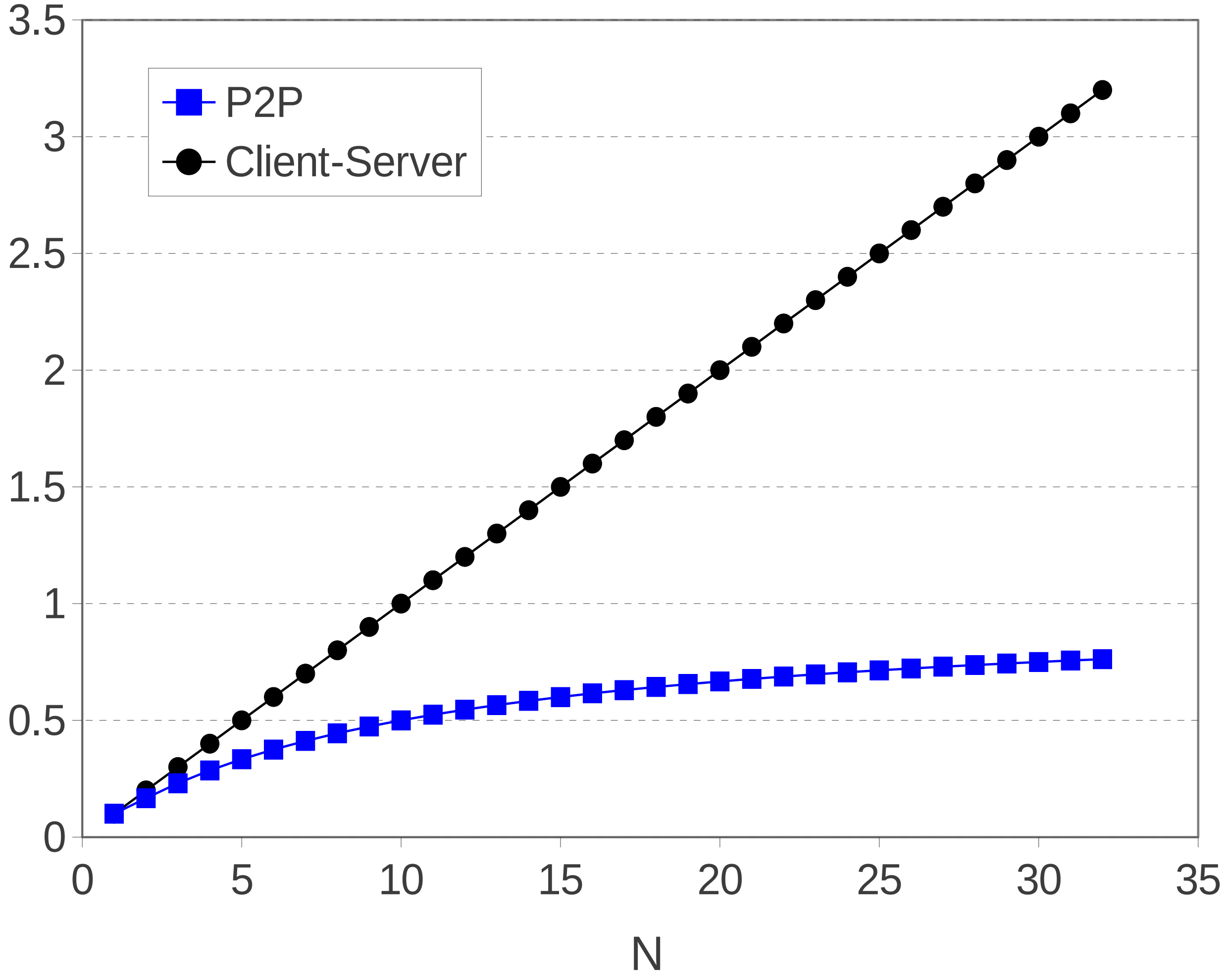
$$D_{P2P} > \max\{F/u_s, F/d_{min}, NF/(u_s + \Sigma u_i)\}$$

increases linearly in *N* …

… but so does this, as each peer brings service capacity

Client-server vs. P2P: example

client upload rate = u, F/u = 1 hour, $u_s = 10u$, $d_{min} \geq u_s$



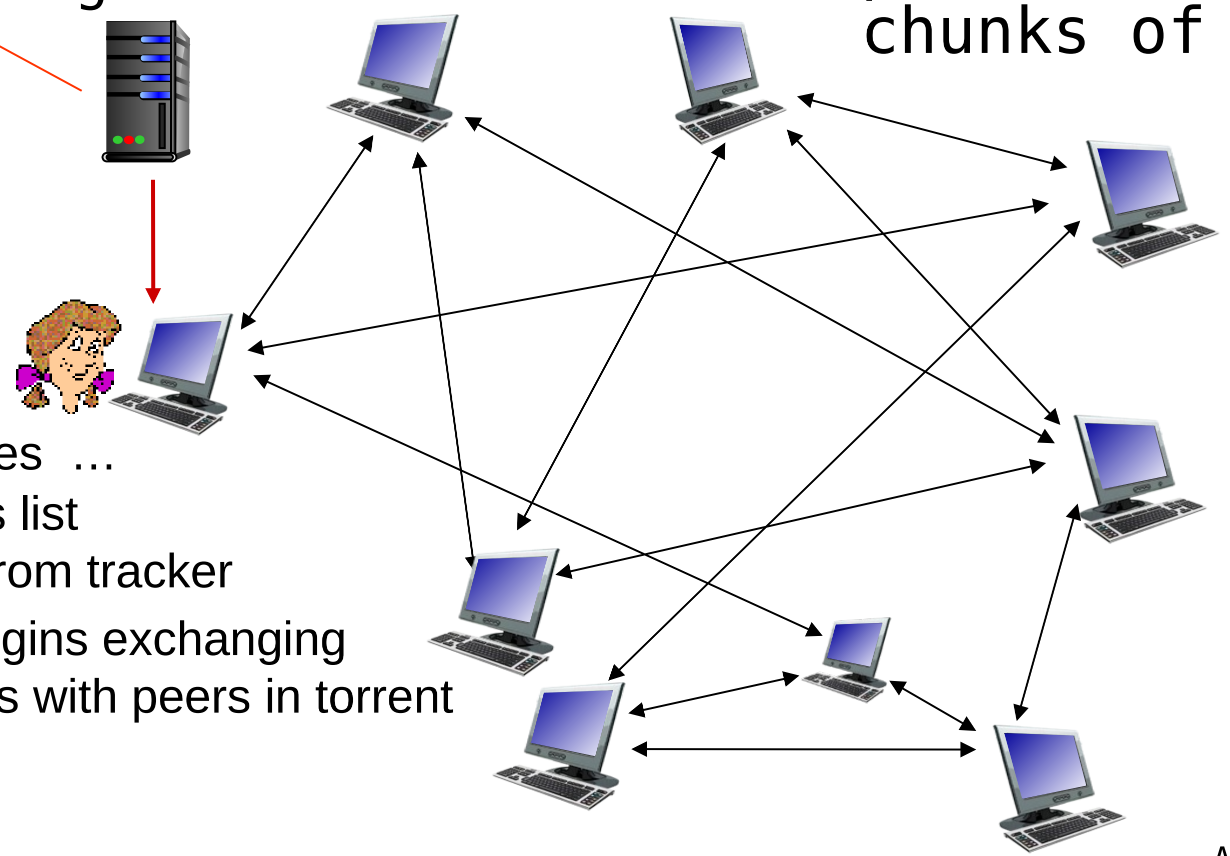
Sistema escalável: crescimento do tempo é sublinear em N

P2P file distribution: BitTorrent

- file divided into 256Kb chunks → 1GB file = 4000 chunks
- peers in torrent send/receive file chunks

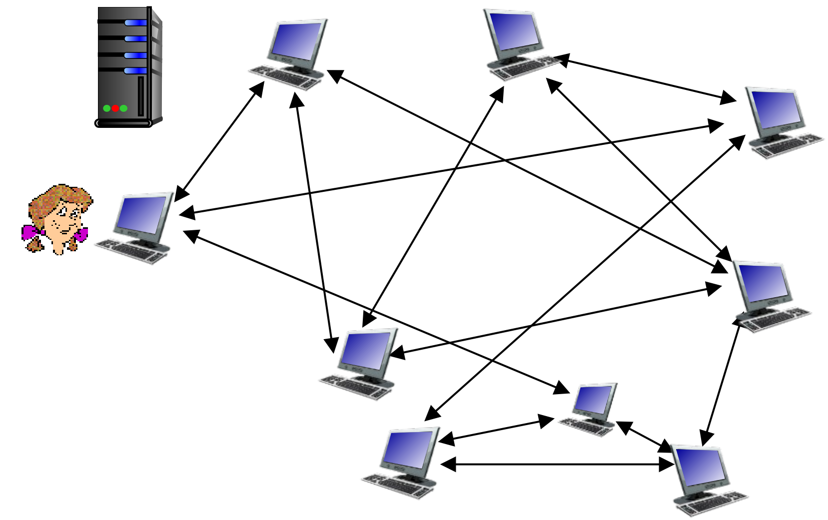
tracker: tracks peers participating in torrent

torrent: group of peers exchanging chunks of a file



Alice arrives …
… obtains list of peers from tracker
… and begins exchanging file chunks with peers in torrent

P2P file distribution: BitTorrent



- **Key idea**: while downloading a chunk peer uploads other chunks to other peers
- as soon as chunk is downloaded, chunk can be uploaded to other peers

- **Three fundamental questions**
- To which peers connect?
  - cannot connect to all if swarm is large
- Which chunks to request for download?
  - if peer has many chunks available
- To which peers upload?
  - many peers may request chunks

BitTorrent: requesting, sending file chunks

requesting chunks:

- at any given time, different peers have different subsets of file chunks
- periodically, Alice asks each peer for list of chunks that they have
- Alice requests missing chunks from peers, ***rarest first***

sending chunks: tit-for-tat

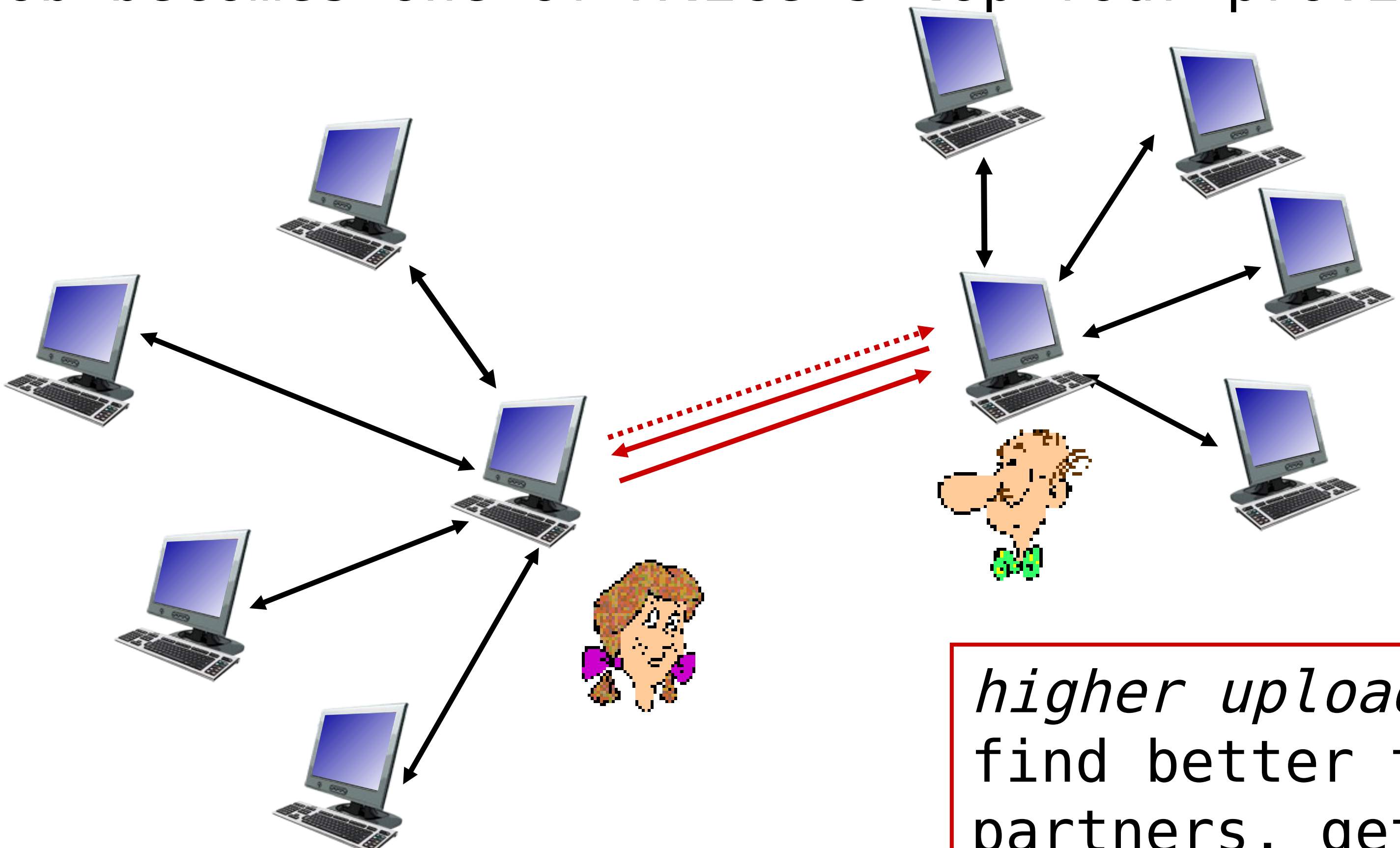
- Alice sends chunks to those four peers currently sending her chunks *at highest rate*
  - other peers are choked by Alice (do not receive chunks from her)
  - re-evaluate top 4 every10 secs
- every 30 secs: randomly select another peer, starts sending chunks
  - "optimistically unchoke" this peer
  - newly chosen peer may join top 4

BitTorrent: tit-for-tat

(1) Alice "optimistically unchokes" Bob

(2) Alice becomes one of Bob's top-four providers; Bob reciprocates

(3) Bob becomes one of Alice's top-four providers



higher upload rate: find better trading partners, get file faster !

Distributed Hash Table (DHT)

- Hash table
- DHT paradigm
- Circular DHT and overlay networks
- Peer churn

Simple Database

Simple database with (key, value) pairs:

- key: human name; value: social security #

Key	Value
John Washington	132-54-3570
Diana Louise Jones	761-55-3791
Xiaoming Liu	385-41-0902
Rakesh Gopal	441-89-1956
Linda Cohen	217-66-5609
…….	………
Lisa Kobayashi	177-23-0199

- key: movie title; value: IP address

Hash Table

- More convenient to store and search on numerical representation of key
- use a *hash function*
- key = hash(original key)

Original Key	Key	Value
John Washington	8962458	132-54-3570
Diana Louise Jones	7800356	761-55-3791
Xiaoming Liu	1567109	385-41-0902
Rakesh Gopal	2360012	441-89-1956
Linda Cohen	5430938	217-66-5609
…….		………
Lisa Kobayashi	9290124	177-23-0199

Distributed Hash Table (DHT)

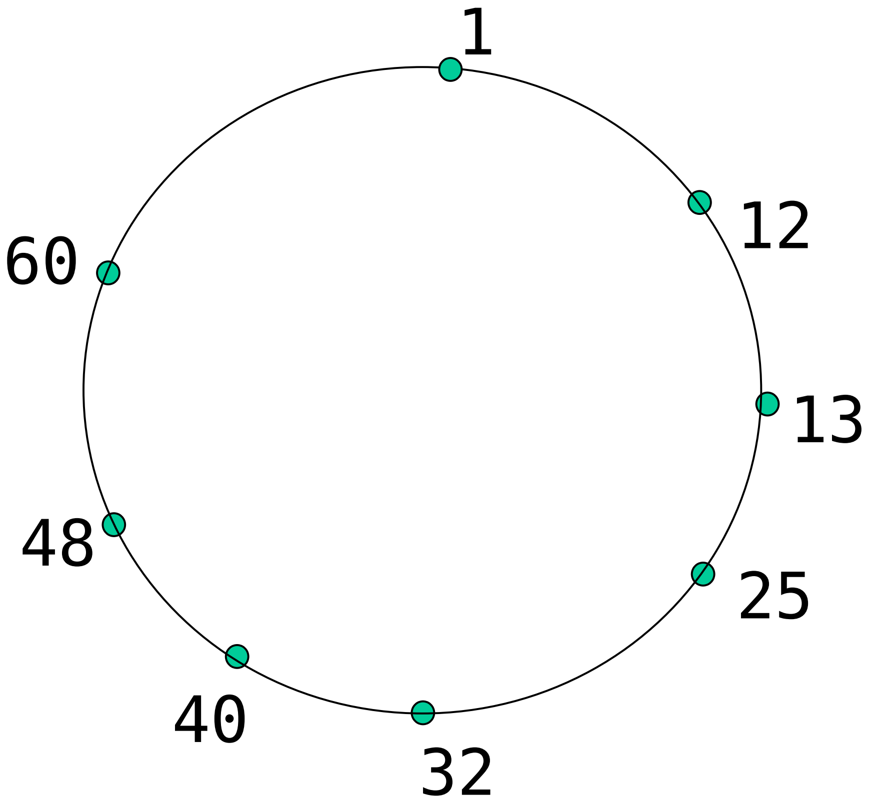
- **Key idea**: distribute (key, value) pairs over set of peers
  - evenly distributed over peers
- Any peer can query DHT with a key
  - DHT returns value for the key
  - to resolve query, small number of messages exchanged among peers
- Each peer only knows about a small number of other peers
- Robust to peers coming and going (churn)

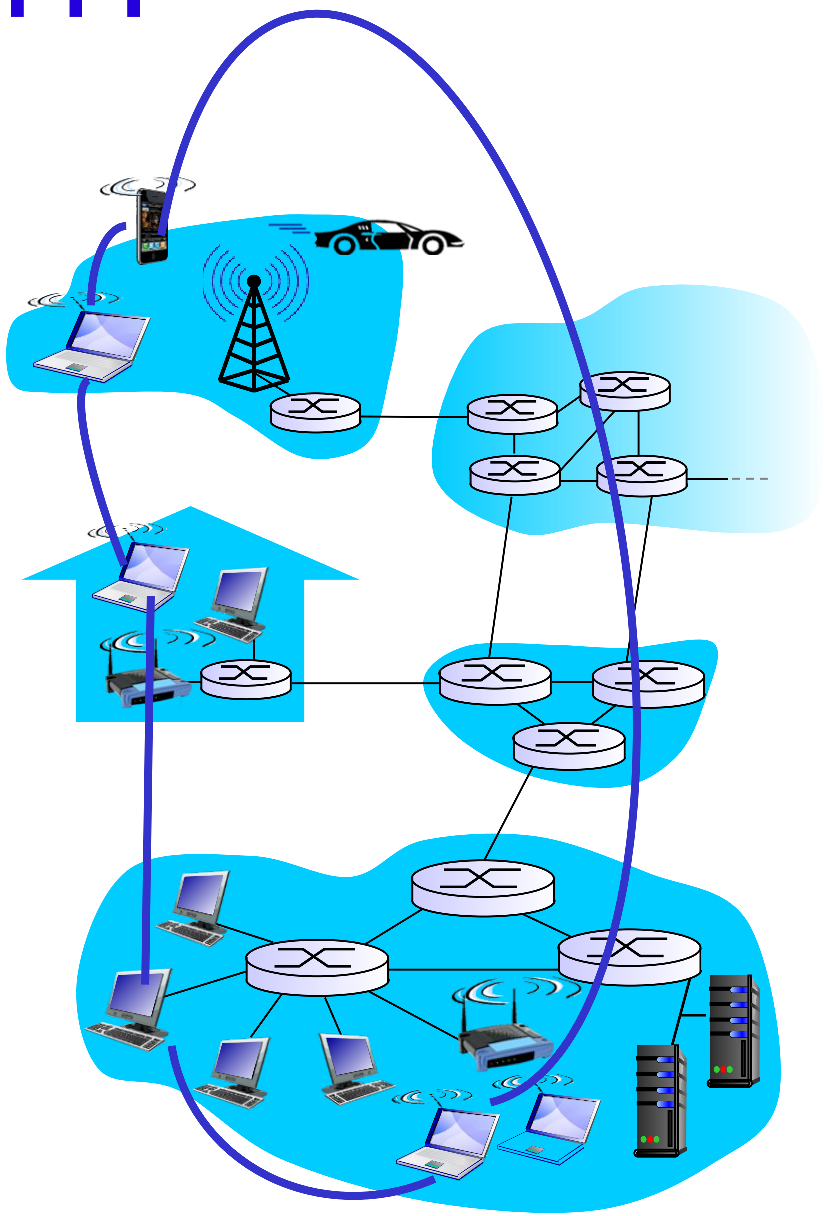
Assign key-value pairs to peers

- **Idea:** map peers to key space; assign key-value pair to the peer that has the *closest* ID.
- convention: closest is the *immediate successor* of the key.
- e.g., ID space {0,1,2,3,…,63}
- suppose 8 peers: 1,12,13,25,32,40,48,60
  - If key = 35, then assigned to peer 40
  - If key = 60, then assigned to peer 60
  - If key = 61, then assigned to peer 1

Circular DHT

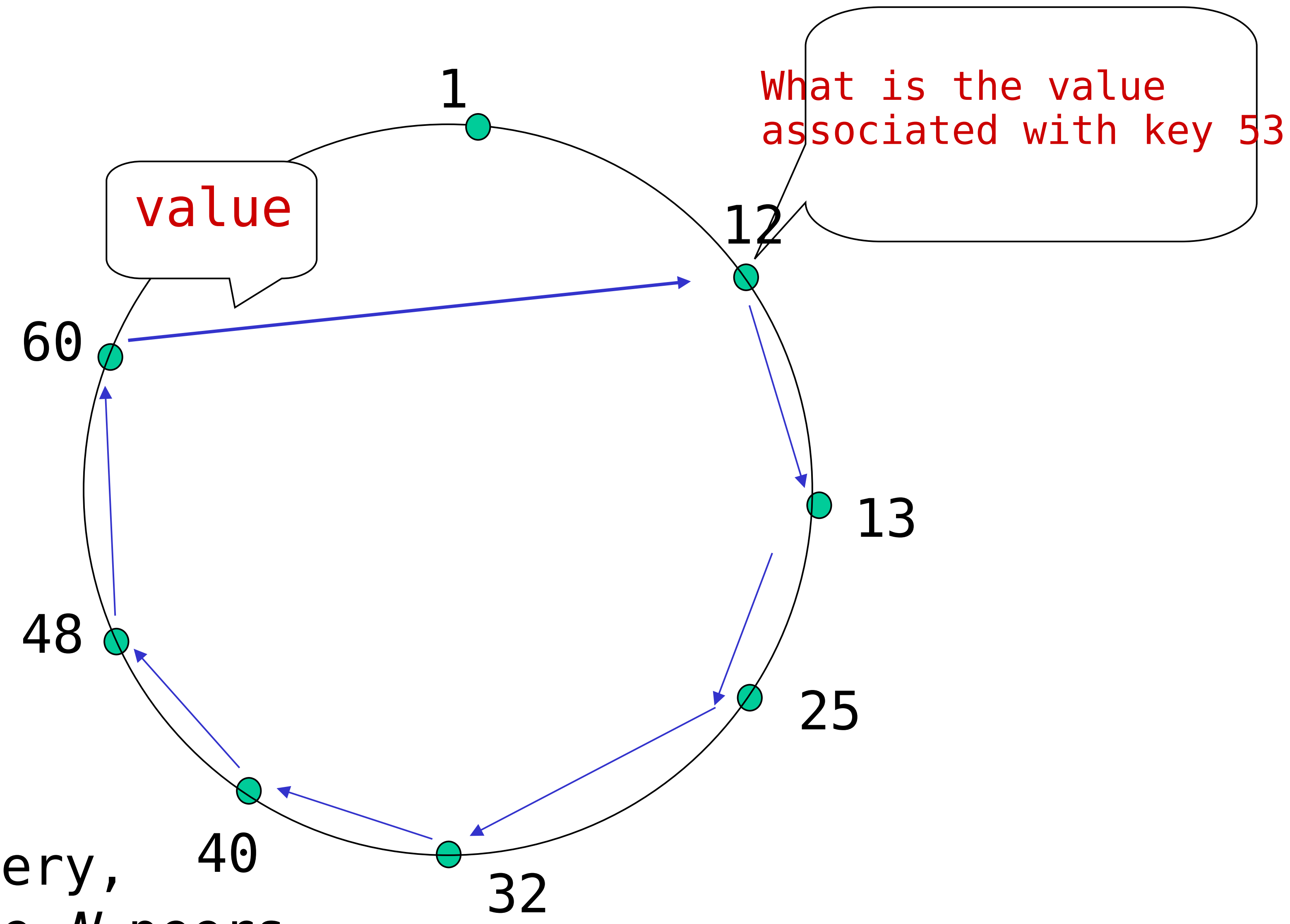
- each peer *only* aware of immediate successor and predecessor.





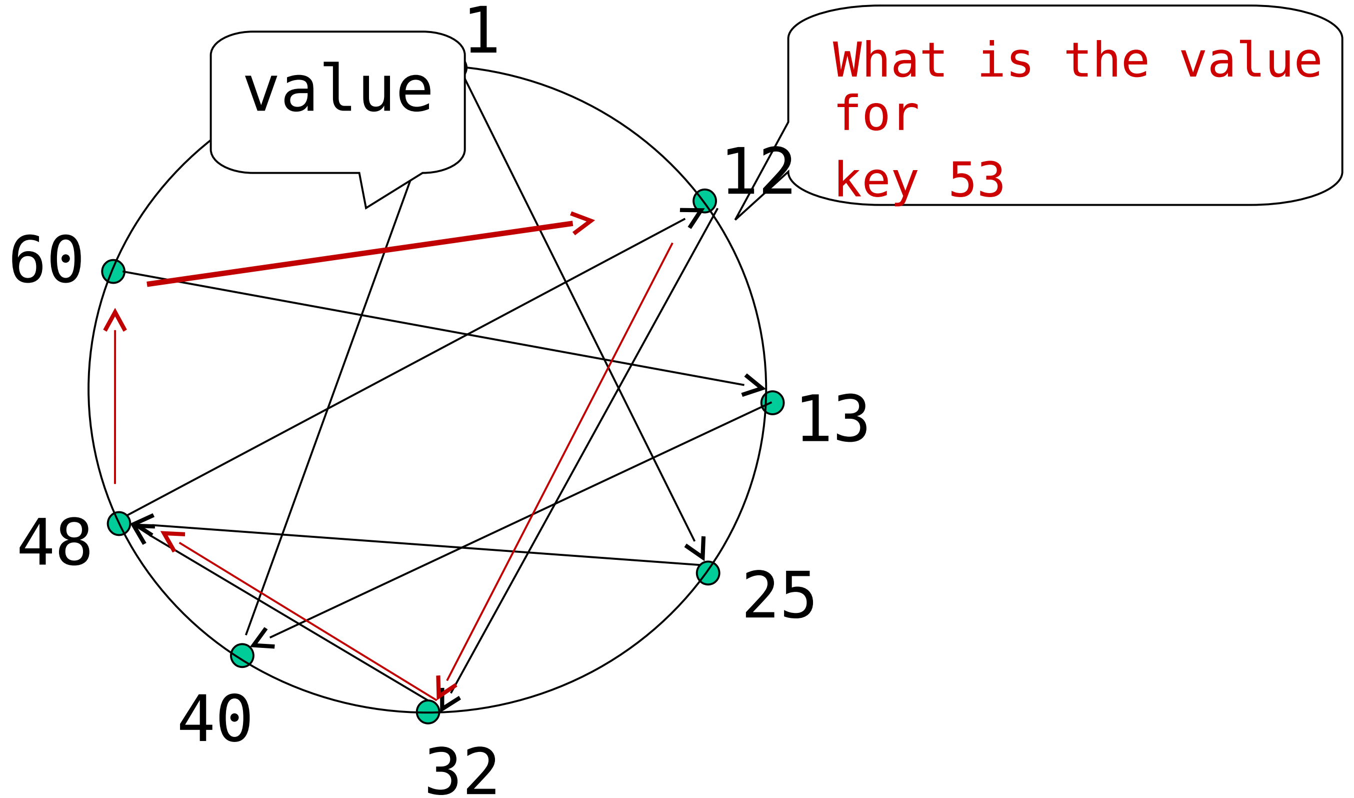
"overlay network"

Resolving a query



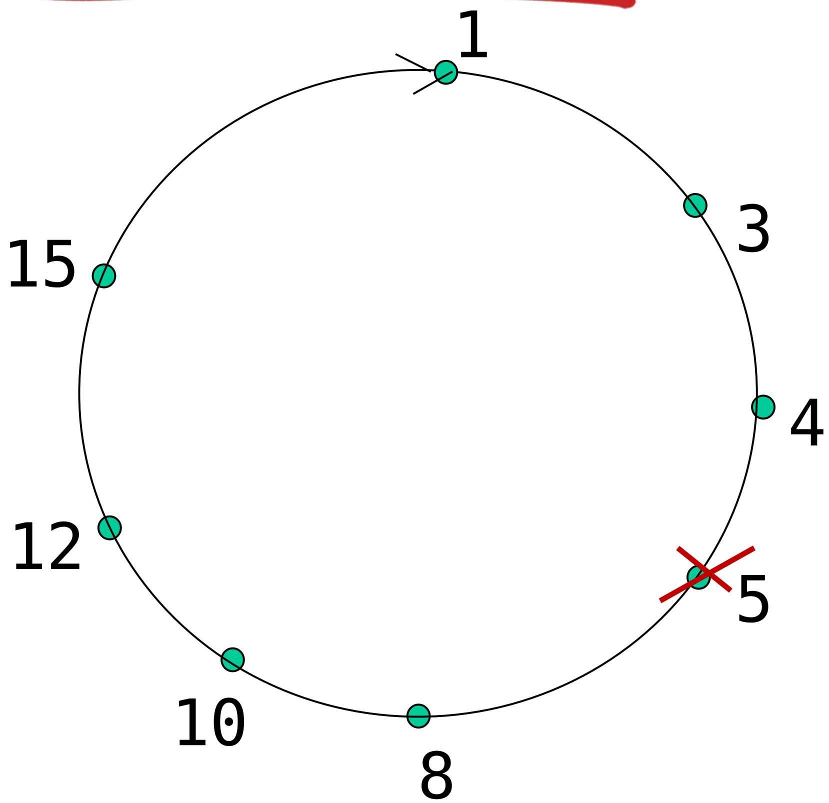
O(N) messages
to resolve query,
when there are *N* peers

Circular DHT with shortcuts



- each peer keeps track of IP addresses of predecessor, successor, and short cuts
- reduced from 6 to 3 messages
- possible to design shortcuts with *O(log N)* neighbors, *O(log N)* messages in query (Chord and CAN are examples)

Peer churn



example: peer 5 abruptly leaves!

handling peer churn:

- peers may come and go (churn)
- each peer knows address of its two successors
- each peer periodically pings its two successors to check aliveness
- if immediate successor leaves, choose next successor as new immediate successor

- peer 4 detects peer 5’s departure; makes 8 its immediate successor
- 4 asks 8 who its immediate successor is; makes 8’s immediate successor its second successor