

Reti di Elaboratori

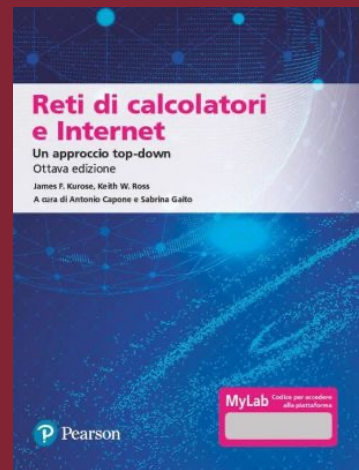
Livello di Applicazione, P2P



SAPIENZA
UNIVERSITÀ DI ROMA

Alessandro Checco

alessandro.checco@uniroma1.it



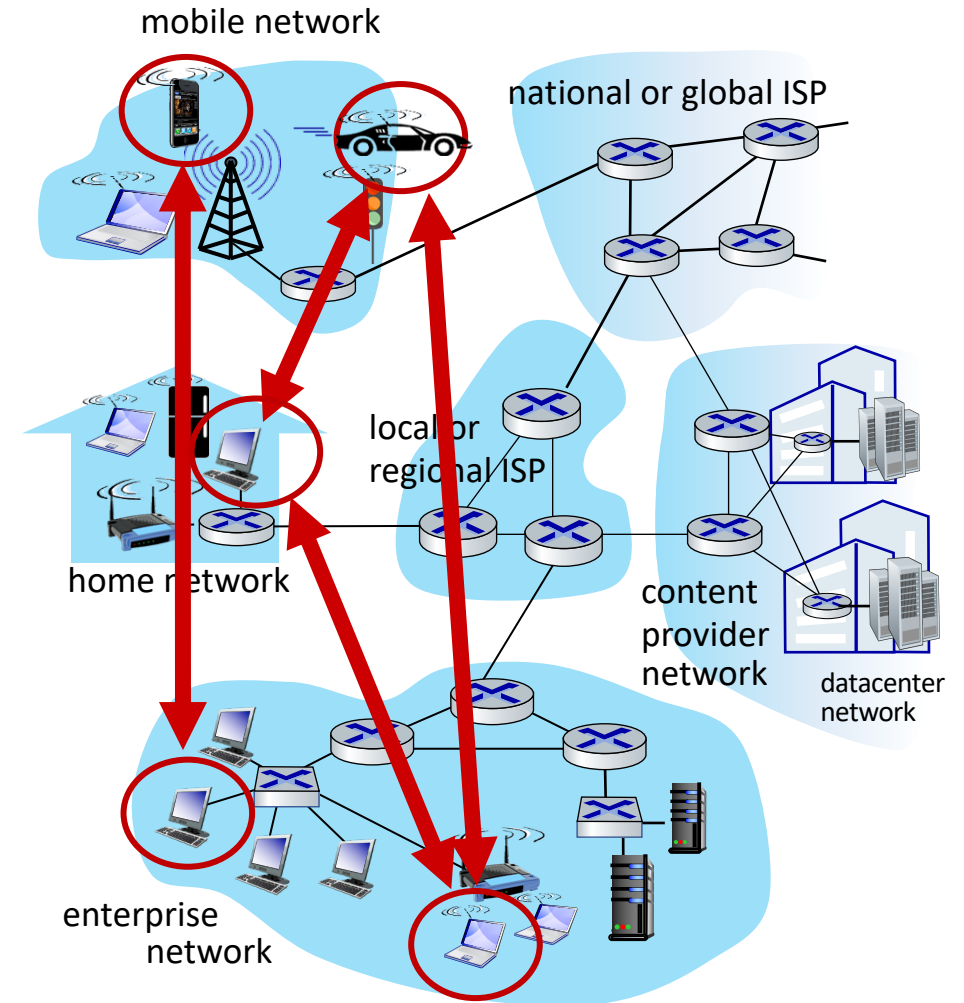
Capitolo 2

Livello di applicazione: sommario

- Principi delle applicazioni di rete
- Web e HTTP
- Posta elettronica, SMTP, IMAP
- FTP
- Domain Name System: DNS
- Applicazioni P2P
- streaming video e content distribution networks
- programmazione socket con UDP e TCP

Paradigma peer-to-peer

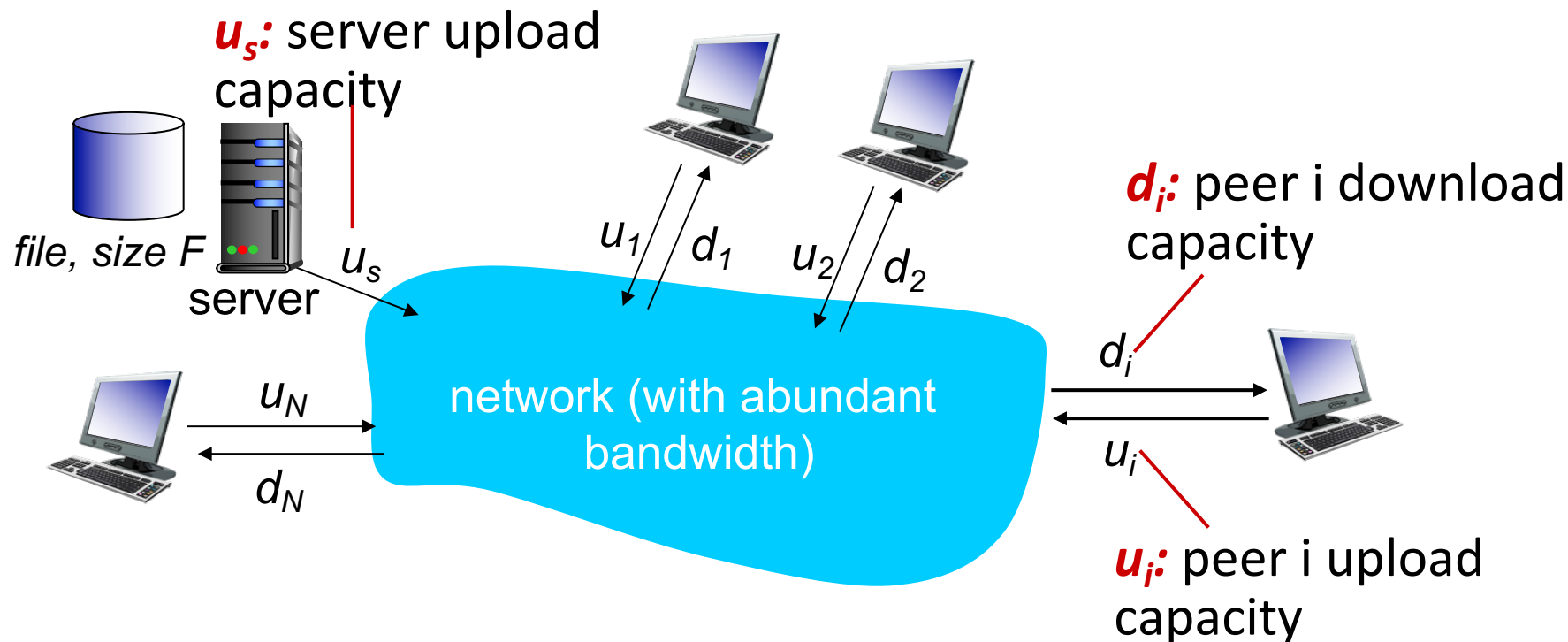
- *nessun* server è sempre attivo
- qualsiasi peer può comunicare direttamente
- i peer richiedono (e forniscono) servizi da altri peer
 - *autoscalabilità*: nuovi peer incrementano la capacità di servizio, nonché nuove richieste di servizio
- i peer sono connessi in modo intermittente con indirizzi IP variabili
 - gestione complessa
- esempio: condivisione di file P2P (BitTorrent), streaming (KanKan), VoIP (Skype)



Distribuzione dei file: client-server vs P2P

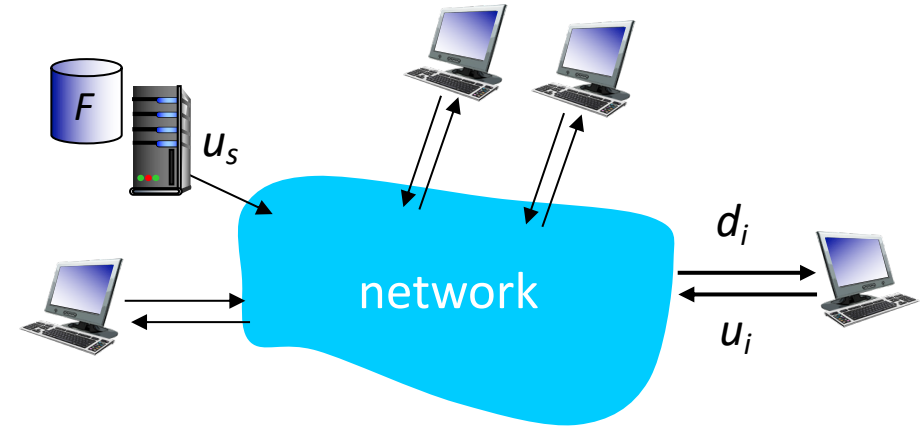
D : quanto tempo occorre per distribuire il file (dimensione F) da un server a N peer?

- la capacità di upload/download peer è una risorsa limitata



Tempo di distribuzione del file: client-server

- **trasmissione server:** deve inviare sequenzialmente N copie del file:
 - tempo per inviare una copia: F/u_s
 - tempo per inviare N copie: NF/u_s
- **client:** ogni client deve scaricare una copia del file
 - d_{min} = velocità minima di download
 - tempo peggiore di download: F/d_{min}



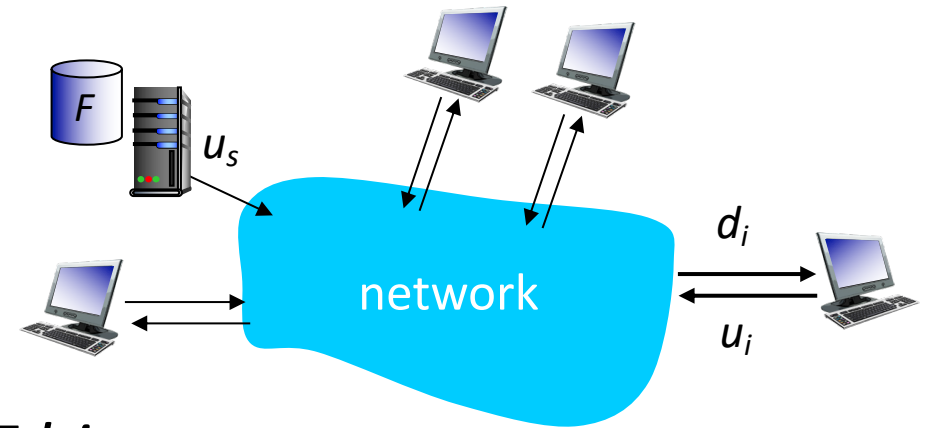
*tempo per distribuire F
a N clienti che utilizzano
approccio client-server*

$$D_{cs} > \max\{NF/u_s, F/d_{min}\}$$

aumenta linearmente con N

Tempo di distribuzione del file: P2P

- **trasmissione server:** deve inviare almeno una copia:
 - tempo per inviare una copia: F/u_s
- **client:** ogni client deve scaricare una copia del file
 - tempo minimo di download: F/d_{min}
- **clienti:** in aggregato devono scaricare NF bit
 - la velocità di caricamento (che limita la velocità massima di download) è $u_s + \sum u_i$



tempo per distribuire F
a N client che utilizzano
approccio P2P

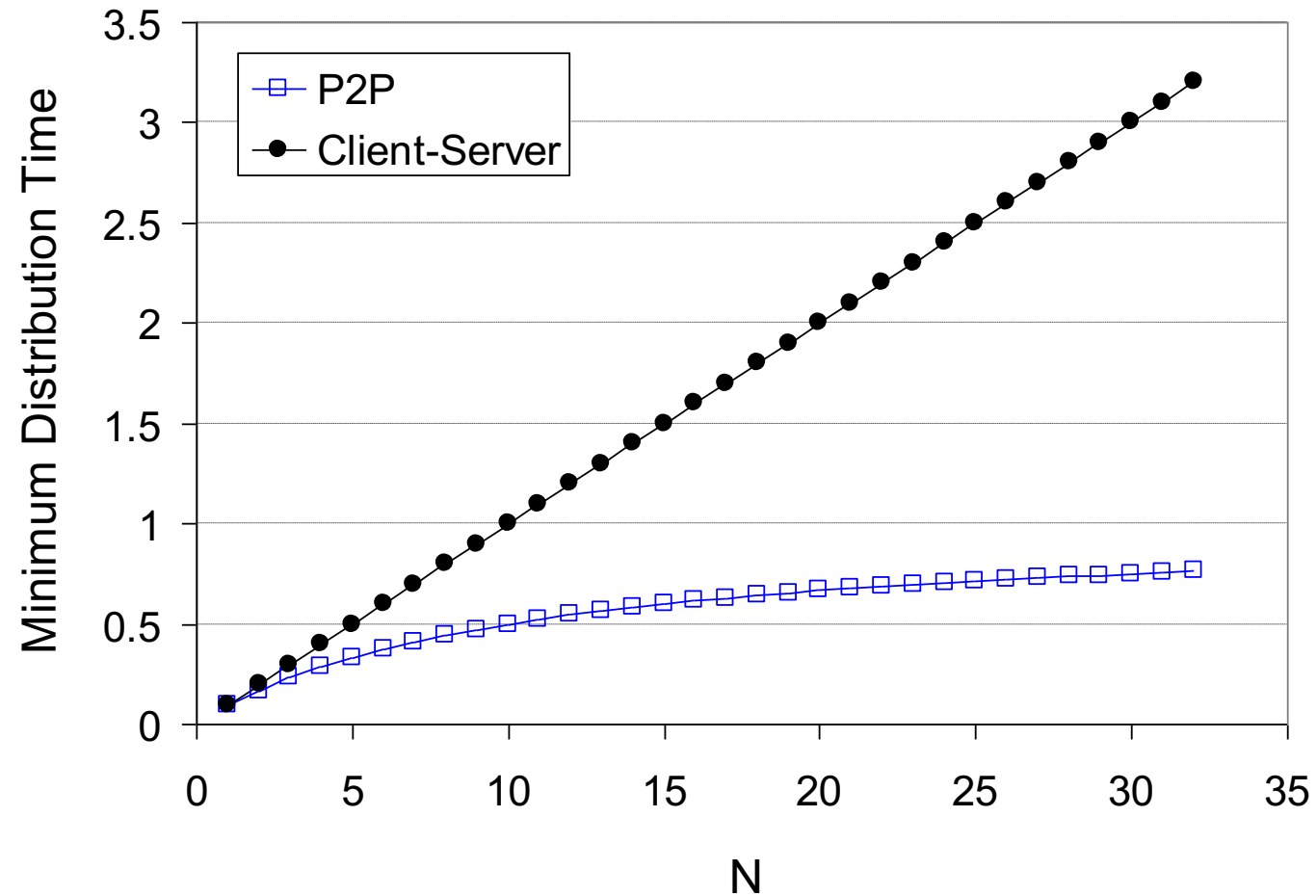
$$D_{P2P} > \max\{F/u_s, F/d_{min}, NF/(u_s + \sum u_i)\}$$

aumenta linearmente in N ...

... ma anche questo, poiché ogni peer apporta capacità di servizio

Client-server vs. P2P: esempio

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

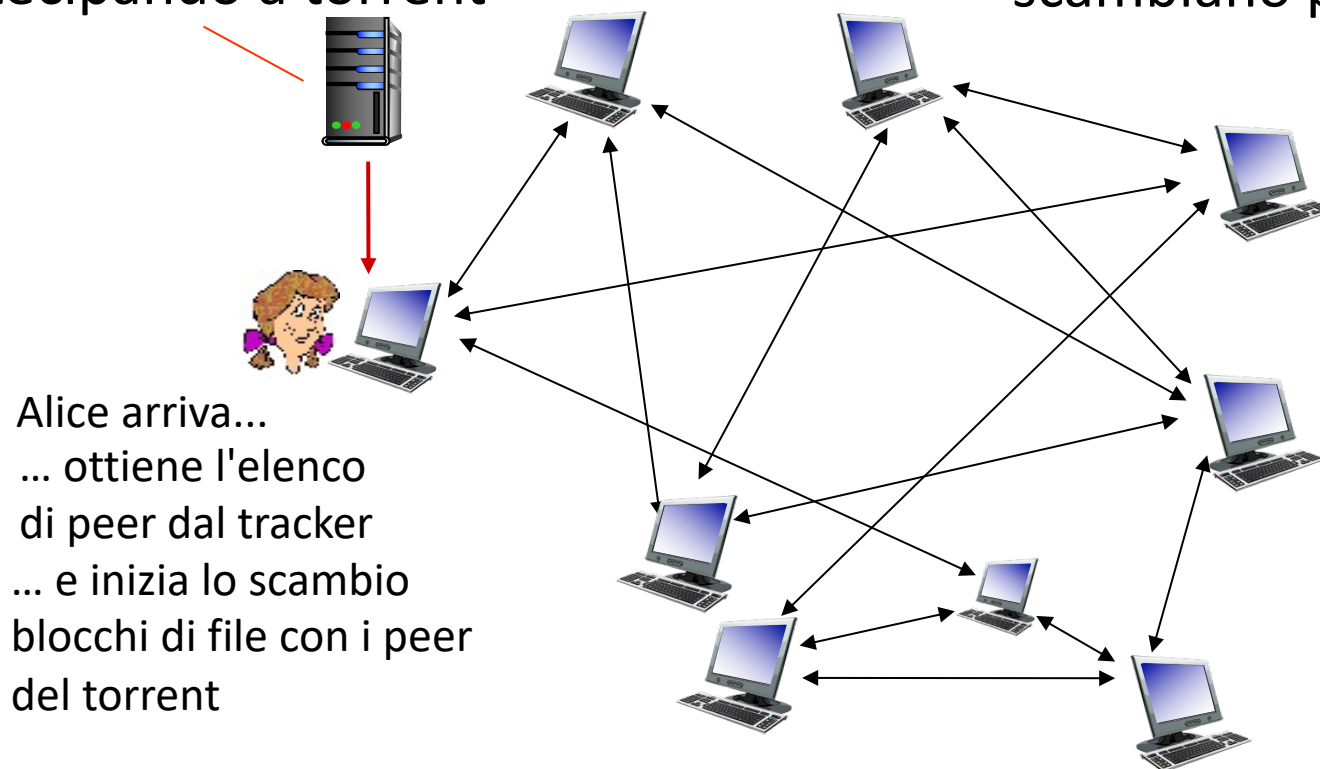


Distribuzione file P2P: BitTorrent

- file diviso in blocchi da 256Kb
- peer mandano e ricevono blocchi

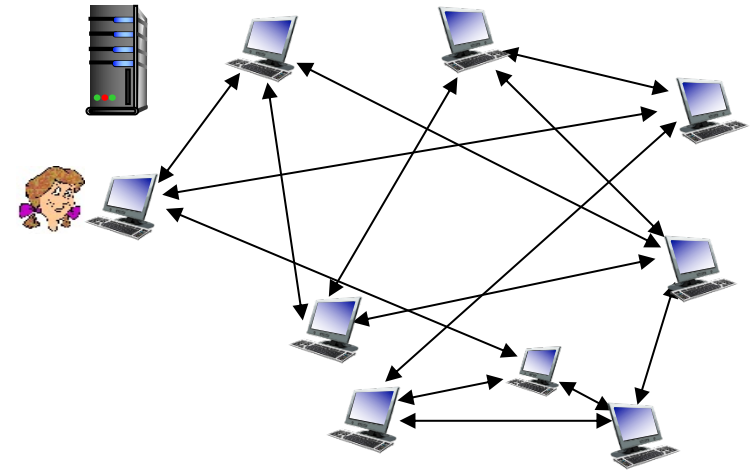
tracker: tiene traccia dei peer partecipando a torrent

torrent: gruppo di peer che si scambiano pezzi di un file



Distribuzione file P2P: BitTorrent

- peer entra in un torrent:
 - non ha blocchi, ma li accumulerà nel tempo da altri peer
 - si registra con il tracker per ottenere l'elenco dei peer, si connette a un sottoinsieme di peer ("vicini")
- durante il download, il peer fa anche upload di blocchi su altri peer
- peer può cambiare peer con i quali scambia blocchi
- *churn*: i peer possono cambiare continuamente e rallentare il sistema
- una volta che peer ha l'intero file, può (egoisticamente) andarsene o (altruisticamente) rimanere nel torrent



BitTorrent: richiesta, invio di blocchi di file

Richiesta di blocchi:

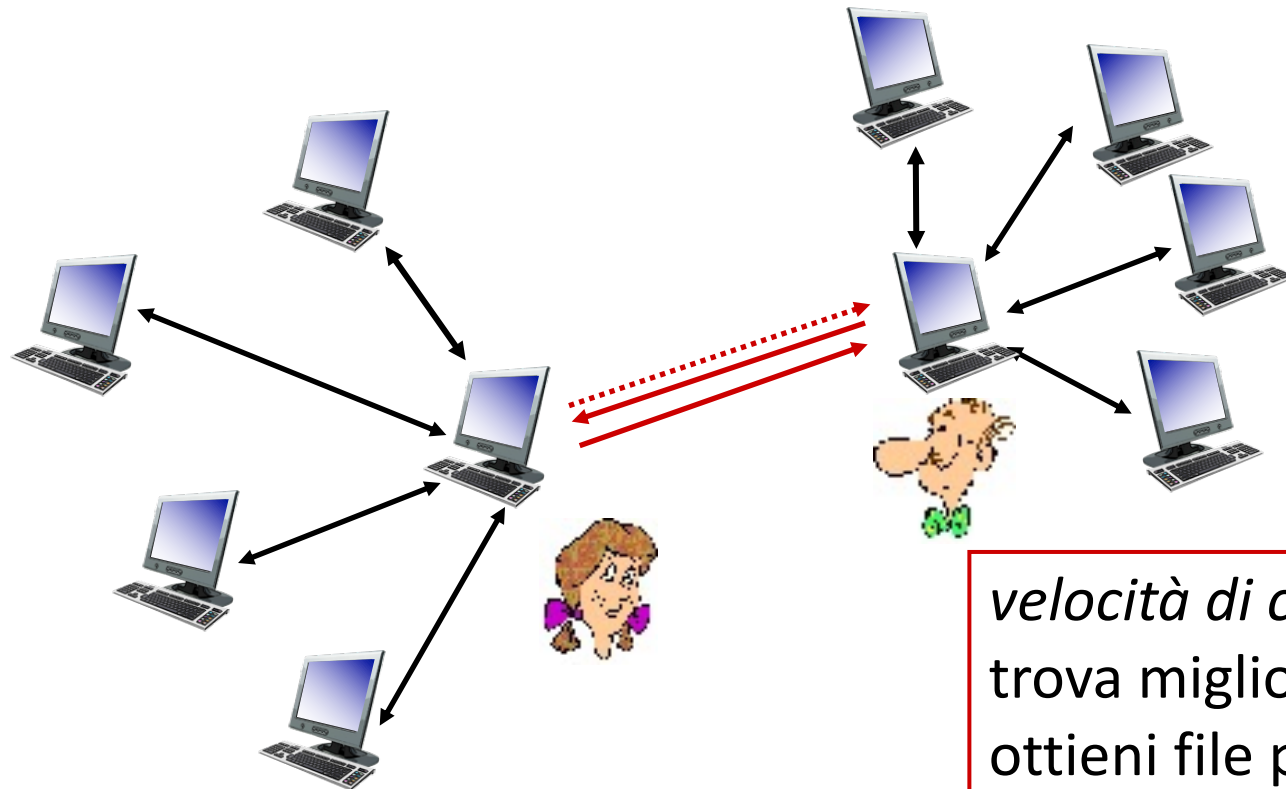
- in un dato momento, peer diversi hanno diversi sottoinsiemi di blocchi di file
- periodicamente, Alice chiede a ciascun peer l'elenco dei blocchi che hanno
- Alice richiede i pezzi mancanti dai peer, partendo dai più rari

Invio di blocchi: tit-for-tat

- Alice invia blocchi ai quattro peer che attualmente inviano i blocchi richiesti *alla velocità* maggiore
 - altri peer vengono «choked» da Alice (non riceveranno blocchi da lei)
 - rivaluta i primi 4 ogni 10 secondi
- ogni 30 secondi: seleziona casualmente un altro peer, inizia a inviare blocchi
 - «optimistic unchoke» per questo peer
 - il peer appena scelto può entrare a far parte della top 4

BitTorrent: tit-for-tat

- (1) Alice "ottimistamente sblocca" Bob
- (2) Alice diventa uno dei primi quattro fornitori di Bob; Bob ricambia
- (3) Bob diventa uno dei primi quattro fornitori di Alice



*velocità di caricamento più elevata:
trova miglior matching di peer,
ottieni file più velocemente!*