Università degli Studi di Roma "Tor Vergata"
Laurea in Informatica

Sistemi Operativi e Reti
(modulo Reti)
a.a. 2024/2025

Livello di applicazione (parte3)

dr. Manuel Fiorelli

manuel.fiorelli@uniroma2.it

https://art.uniroma2.it/fiorelli

Basate sulle slide del libro di testo:
https://gaia.cs.umass.edu/kurose_ross/ppt.php

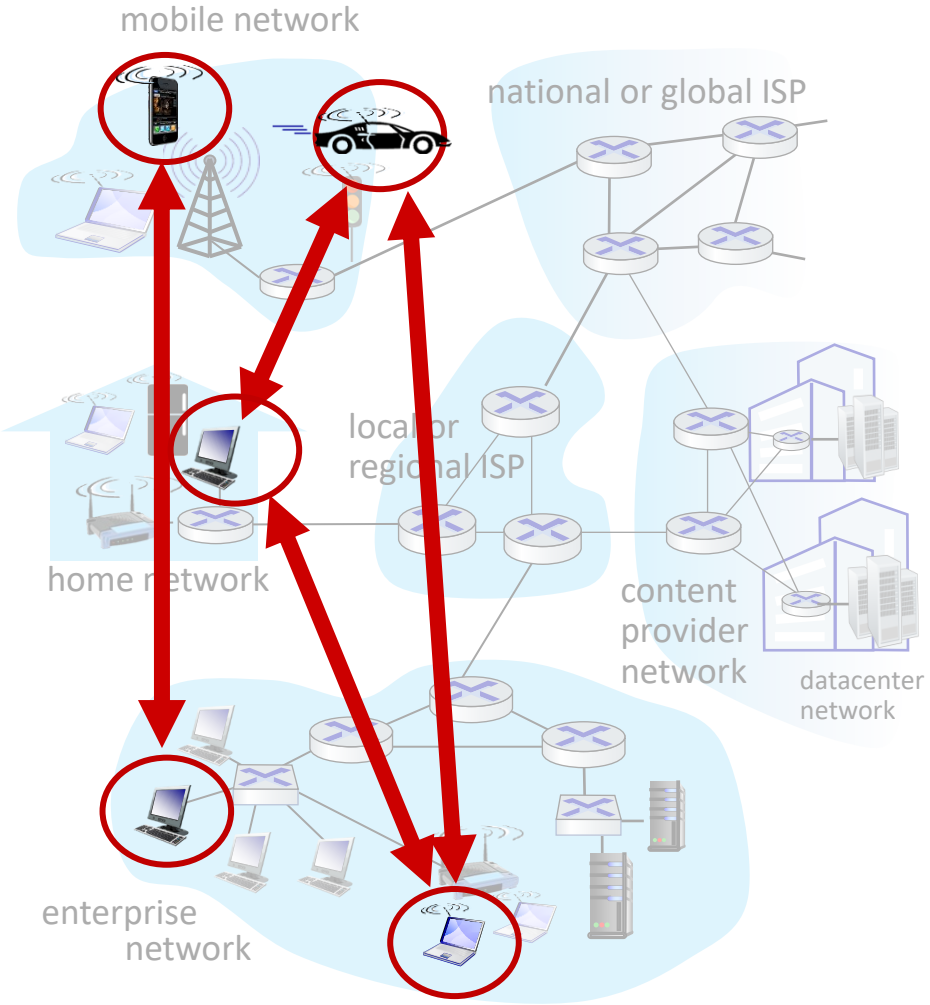
Application Layer: Overview

- Principi delle applicazioni di rete
- Web e HTTP
- E-mail, SMTP, IMAP
- DNS: il servizio di directory di Internet

- Applicazioni P2P
- Streaming video e reti di distribuzione di contenti
- Programmazione delle socket programming con UDP e TCP

Architettura Peer-to-peer (P2P)

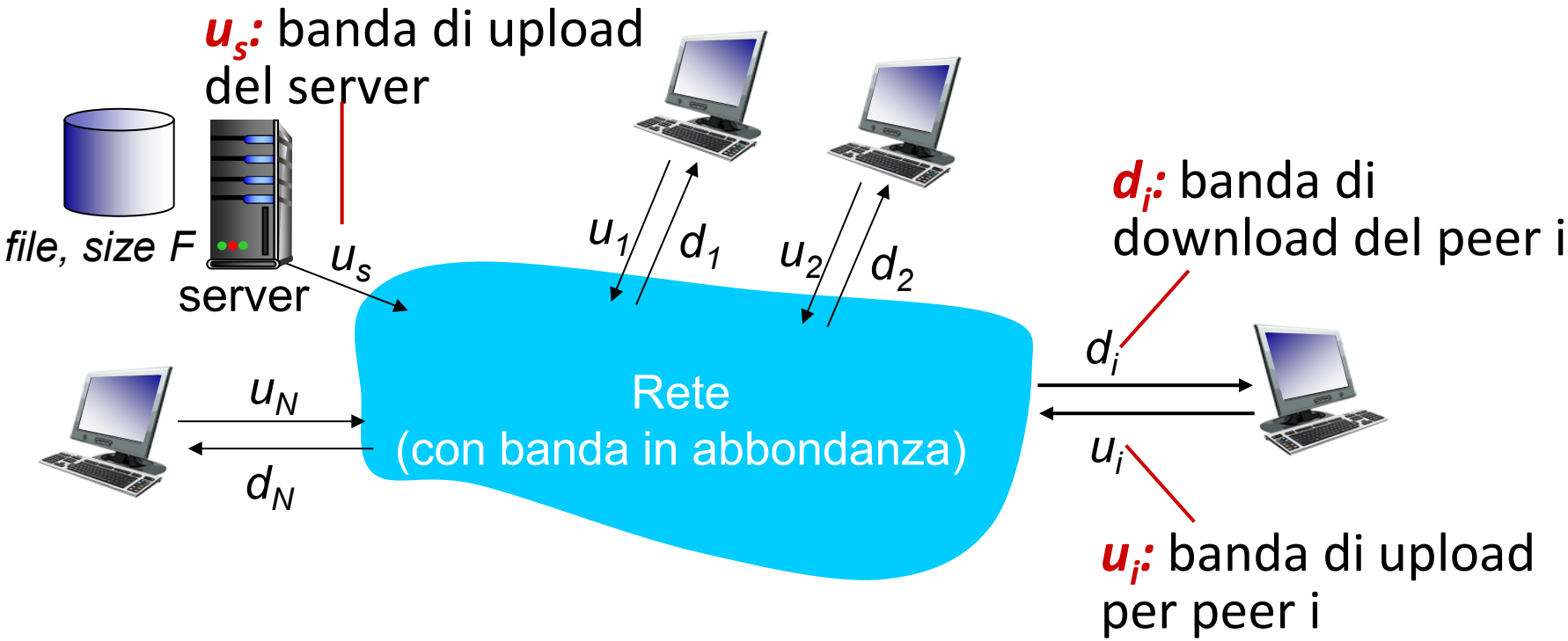
- *nessun* server sempre attivo
- sistemi periferici arbitrari comunicano direttamente
- i peer richiedono un servizio ad altri peer e forniscono un servizio in cambio ad altri peer
  - *scalabilità intrinseca - nuovi peer portano nuova capacità di servizio e nuove richieste di servizio*
- I peer sono connessi a intermittenza e cambiano indirizzo IP
  - gestione complessa
- Esempi: P2P file sharing (BitTorrent), streaming (KanKan), VoIP (Skype)



Distribuzione di file: client-server vs P2P

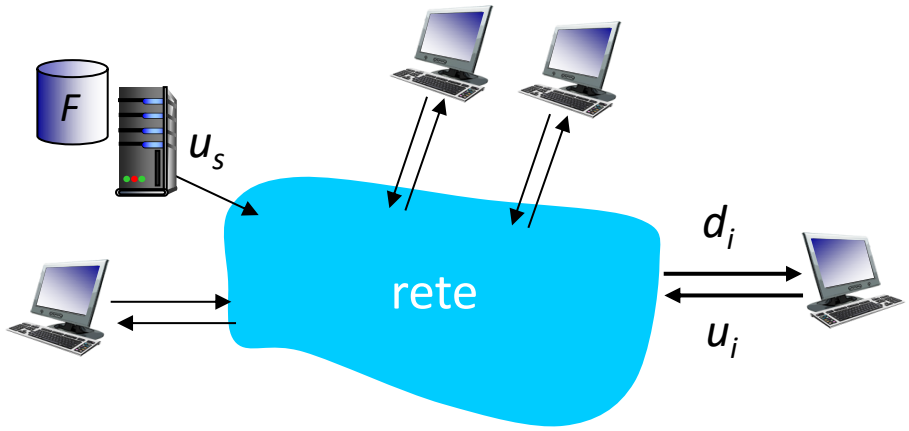
D: quanto tempo per distribuire un file (di dimensione *F*) da un server a *N peer*?

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



File distribution time: client-server

- *trasmissione via server:* deve inviare (caricare) in sequenza N copie di 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} = banda di download più bassa
  - tempo di download per il client con banda minima è almeno: F/d_{min}



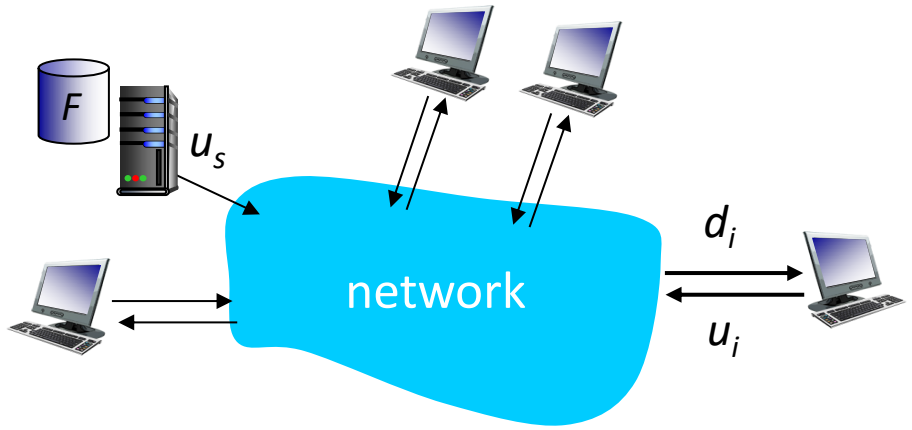
Tempo per distribuire F a N client usando l'approccio client-server

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

aumenta linearmente in N

Distribuzione di file: P2P

- *trasmissione via server:* deve trasmettere almeno una copia del file:
  - tempo per inviare un copia: F/u_s
- *client:* ogni client deve scaricare una copia del file
  - Tempo per il client più lento, almeno F/d_{min}
- I *client:* come aggregato devono scaricare NF bit
  - capacità totale di upload (che limita la massima velocità di download) è $u_s + \Sigma u_i$



Tempo per distribuire F a N client usando l'approccio P2P

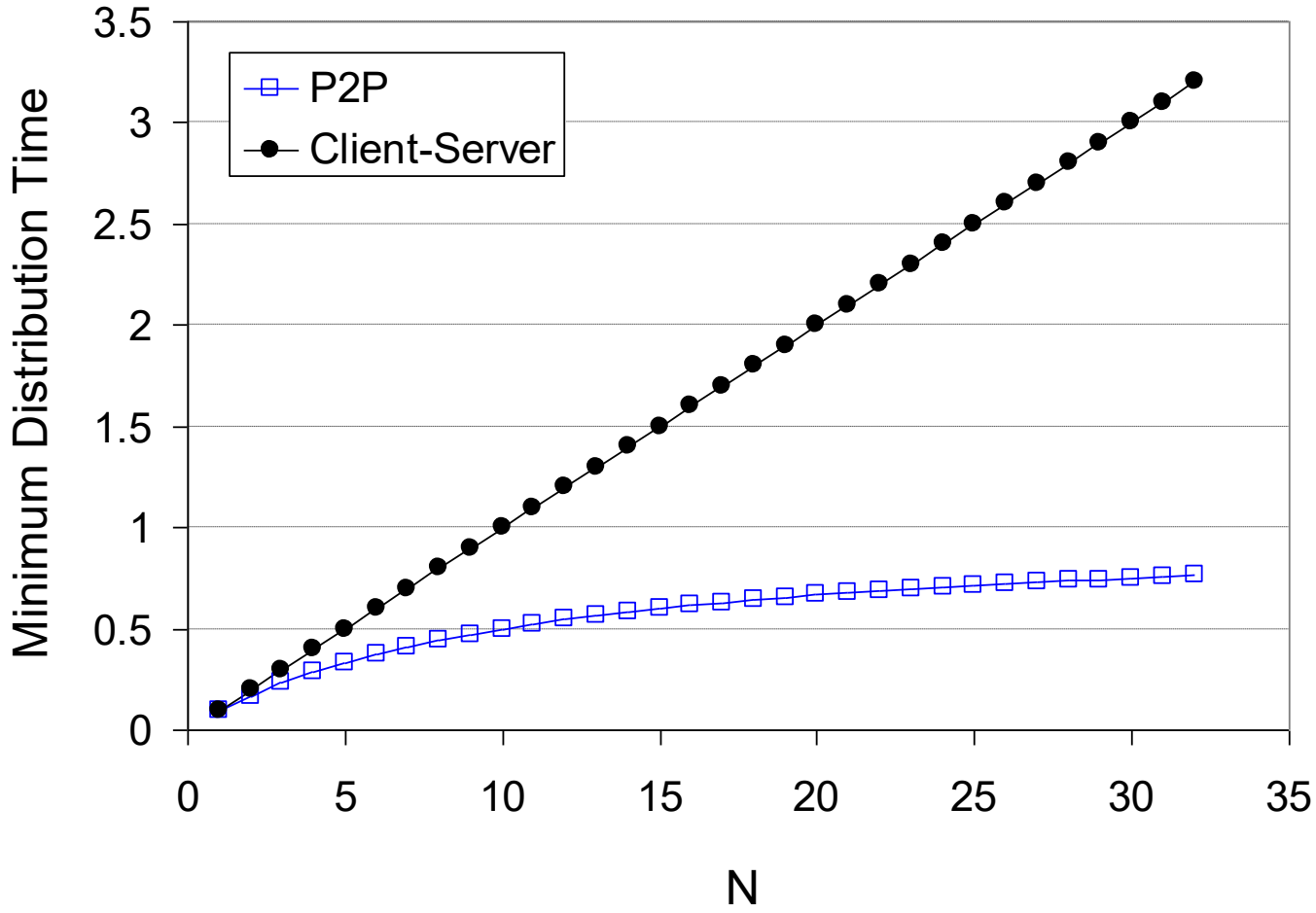
$$D_{P2P} \geq max\{F/u_s, F/d_{min}, NF/(u_s + \Sigma u_i)\}$$

aumenta linearmente in N …

… ma anche questo, dato che ogni peer porta con sé la capacità di servizio

Client-server vs. P2P: example

banda di upload del client = u, F/u = 1 ora, $u_s = 10u$, $d_{min} \geq u_s$

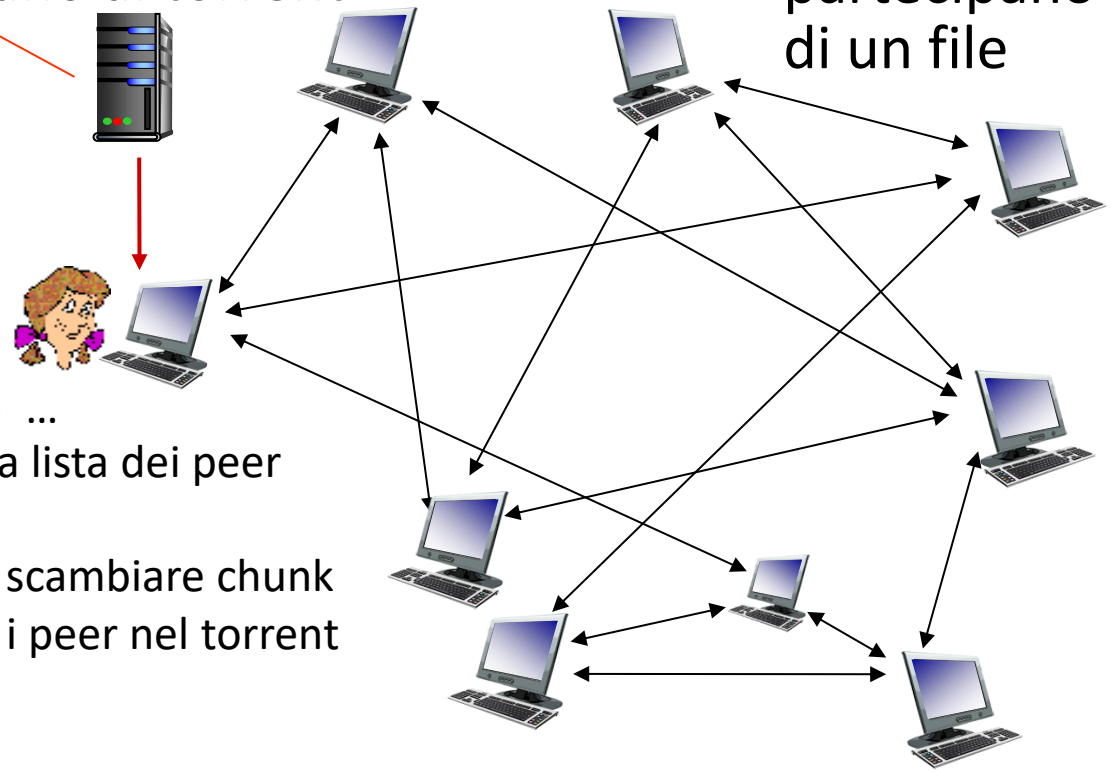


Distribuzione di file P2P: BitTorrent

- file diviso in chunk (parti), in genere di 256 kB
- i peer nel torrent inviano/ricevono chunk del file

tracker: tiene traccia dei peer che partecipano al torrent

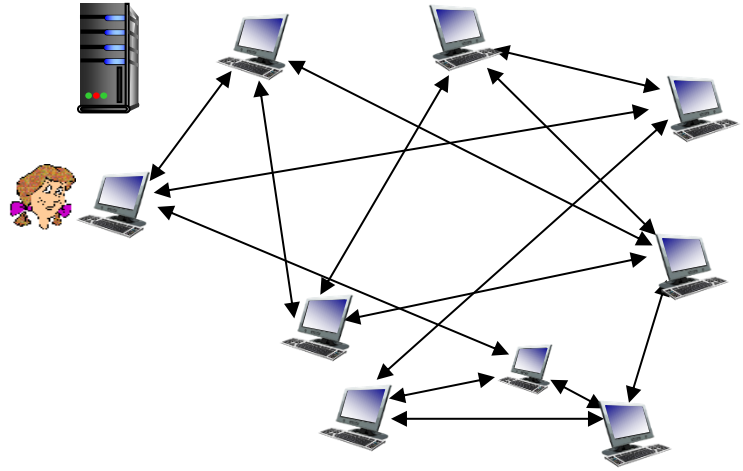
torrent: gruppo di peer che partecipano alla distribuzione di un file



Alice arriva …
… ottiene la lista dei peer dal tracker
… e inizia a scambiare chunk del file con i peer nel torrent

Distribuzione di file P2P: BitTorrent

- Un peer che entra a far parte del torrent:
  - non ha chunk del file, ma li accumulerà nel tempo da altri peer
  - si registra con un tracker, ottenendo la lista di un sottoinsieme dei peer nel torrent (es. 50), stabilisce una connessione con un sottoinsieme di questi, che sono detti peer "vicini" (“neighbors”)
  - informa periodicamente il tracker che è ancora nel torrent



- mentre scarica chunk, un peer invia i chunk già in suo possesso agli altri peer
- un peer può cambiare i peer con cui scambia i chunk
- i peer possono andare e venire
- una volta che un peer ha acquisito l'intero file, può (egoisticamente) lasciare il torrent oppure può (altruisticamente) rimanere nel torrent (come *seeder*)

BitTorrent: richiesta e invio di chunk di file

Richiesta di chunk:

- in ogni momento, peer diversi hanno sottoinsiemi diversi di chunk
- periodicamente, Alice chiede ai peer vicini l'elenco dei chunk in loro possesso
- Alice richiede ai peer i chunk mancanti, adottando la strategia del **rarest first** (**"**prima i più rari"): uniformando la distribuzione dei chunk, migliora la disponibilità globale e aumenta le possibilità di scambio (maggiore throughput)
- Un peer appena entrato può chiedere un blocco in modo casuale (perché vuole avere il prima possibile un blocco da condividere); mentre, quando sta per completare il file, può adottare la strategia end game e richiedere lo stesso blocco a più peer simultaneamente (cancellando le richieste pendenti appena appena riceve un blocco)

Invio di chunk: tit-for-tat ("pan per focaccia")

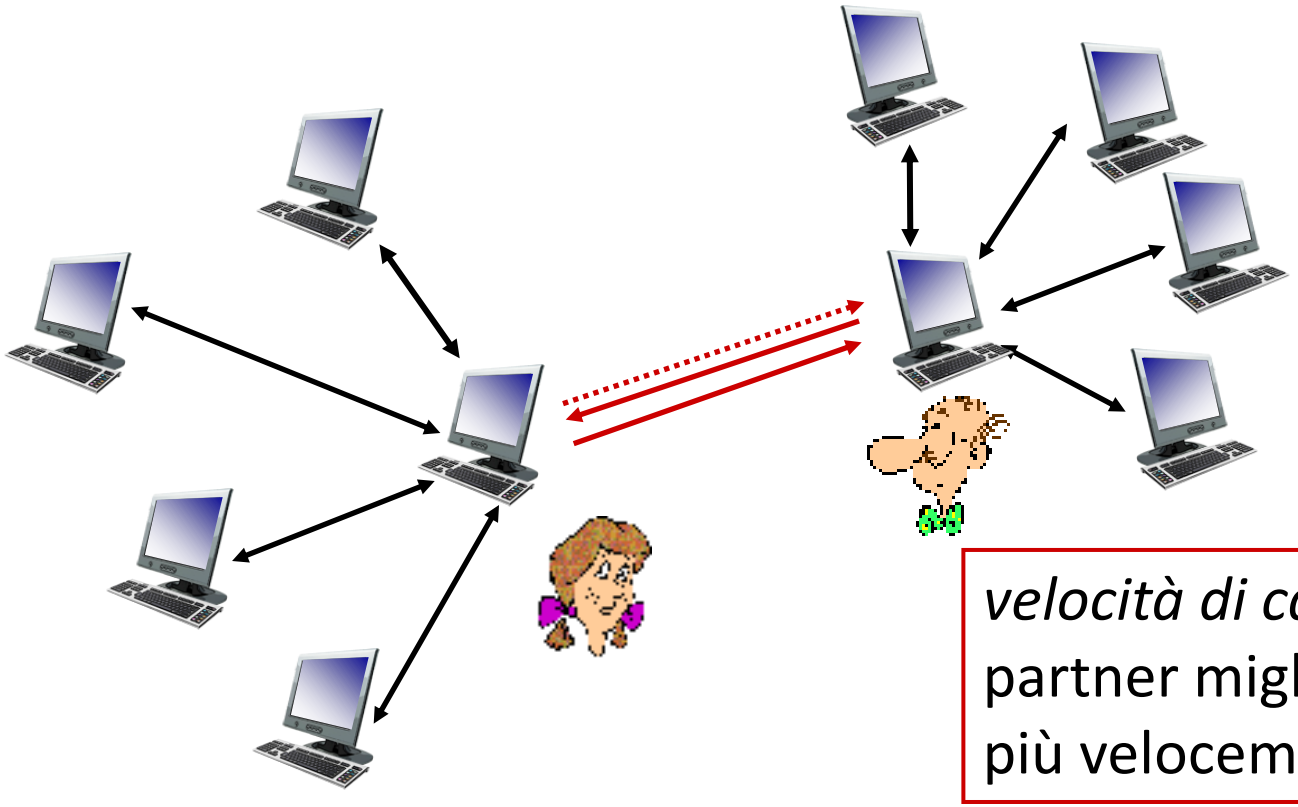
- Alice invia i chunk ai quattro peer vicini che attualmente le inviano i chunk *alla velocità più alta*
  - altri peer sono detti choked ("soffocati" o "limitati") (non ricevono chunk da Alice)
  - rivaluta i primi 4 posti ogni 10 secondi
- ogni 30 secondi: seleziona in modo casuale un vicino, inizia a inviare chunk
  - questo peer è detto “optimistically unchoked” (“non limitato/soffocato in maniera ottimistica”)
  - il nuovo peer scelto può entrare nella top 4

BitTorrent: tit-for-tat

(1) Alice scelte Bob come “optimistically unchoked”

(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: trovare partner migliori per gli scambi, ottenere file più velocemente!