Università di Roma Tor Vergata
Corso di Laurea triennale in Informatica

Sistemi operativi e reti

A.A. 2020-2021

Pietro Frasca

Parte II: Reti di calcolatori

Lezione 11 (35)

Martedì 13-04-2021

Video Internet

- Lo streaming di video preregistrati rappresenta oggi la maggior parte del traffico residenziale.
- Nelle applicazioni multimediali audio/video, il contenuto da trasmettere in streaming è il video preregistrato, come un film, un evento sportivo preregistrato o video prodotti dagli utenti internet e inviati a siti come, ad esempio, YouTube.
- Questi file multimediali vengono memorizzati su server e gli utenti inviano richieste per visualizzarli.
- Molte società Internet oggi forniscono video in streaming, tra cui Netflix, YouTube (Google), Amazon.
- Per poter trasmettere enormi quantità di flussi video agli utenti distribuiti in tutto il mondo, quasi tutte le principali società di streaming video fanno uso di CDN.

- Un video è una sequenza di immagini, in genere viene visualizzato a una frequenza costante, ad esempio, a 25 o 30 immagini al secondo.
- Un'immagine non compressa, codificata in modo digitale, è costituita da una matrice di pixel. I file multimediali occupano molto spazio e pertanto vengono compressi.
- Gli algoritmi di compressione standardizzati di oggi possono comprimere un video in molti livelli di qualità, in base al bit rate desiderato. Maggiore è il bit rate, migliore è la qualità dell'immagine.
- I video Internet compressi in genere variano da 100 kbps per video di bassa qualità a oltre 2 Mbps per lo streaming di film ad alta definizione; lo streaming 4K può richiedere un bitrate di oltre 10 Mbps.
- Pertanto, più è elevato il bit rate tanto più è grande lo spazio necessario per la loro memorizzazione e tanto più elevato è il traffico che si produce nel trasmetterlo. Ad esempio, un video da 2 Mbps con una durata di 1 ora occupa circa 1 gigabyte di spazio su memoria secondaria.

- Un parametro importante per la misura di qualità per lo streaming video è il Throughput tra server trasmittente e client ricevente.
- Per avere una visione fluida, la rete deve fornire un throughput medio per l'applicazione multimediale che è almeno grande quanto il bit rate del video compresso.
- Per consentire la visione del video più continua possibile a un maggiore numero di utenti, generalmente si creano più versioni dello stesso file video, ognuna a un diverso livello di qualità. Ad esempio, un video potrebbe essere compresso in tre versioni a diverse velocità di 300 kbps, 1 Mbps e 3 Mbps. Gli utenti possono quindi decidere quale versione vogliono vedere in funzione della larghezza di banda disponibile. Gli utenti con connessioni Internet ad alta velocità possono scegliere la versione da 3 Mbps; gli utenti che guardano il video con uno smartphone possono scegliere la versione da 300 kbps.

Streaming HTTP e DASH

- Per la trasmissione di video *on demand* si usa spesso l'HTTP. Nello streaming HTTP, i file video sono memorizzati su un server web come qualsiasi altro tipo di file. Quando un utente vuole vedere un video, l'applicazione client invia un messaggio di richiesta HTTP con il metodo GET, specificando l'URL del video desiderato.
- Il server invia quindi il file video, all'interno di un messaggio di risposta HTTP. La velocità di trasmissione dipende dal percorso, esistente tra client e server, in cui si è instaurata la connessione TCP. In pratica, il troughtput dipende dalle caratteristiche dei collegamenti e dal livello del traffico.
- Sul lato client, i byte dello streaming che man mano arrivano, sono memorizzati in un buffer gestito dall'applicazione client. Quando il numero di byte presenti nel buffer supera una soglia predefinita, l'applicazione client inizia la riproduzione del video. In particolare, l'applicazione acquisisce periodicamente fotogrammi video dal buffer, li decomprime e li visualizza.

- Pertanto, l'applicazione client visualizza gli ultimi frame del video presenti nel buffer.
- Sebbene lo streaming HTTP sia ampiamente usato, tuttavia presenta un grande difetto: tutti i client ricevono la stessa codifica del video, nonostante le grandi differenze di larghezza di banda di cui i numerosi client possono avere a disposizione. Ciò ha portato allo sviluppo di un nuovo tipo di streaming basato su HTTP, detto **DASH** (**Dynamic Adaptive Streaming su HTTP)**.
- Con DASH, il video è codificato in diverse versioni, ciascuna con un bit rate diverso e, di conseguenza, con un livello di qualità diversa. Il client richiede dinamicamente blocchi (chunk) di segmenti video di alcuni secondi di durata. Quando la quantità di larghezza di banda disponibile è elevata, il client seleziona pezzi da una versione di alto livello; e quando la larghezza di banda disponibile è bassa, naturalmente seleziona i blocchi da una versione a basso bit rate.
- Il client seleziona diversi blocchi uno alla volta con messaggi di richiesta HTTP GET.

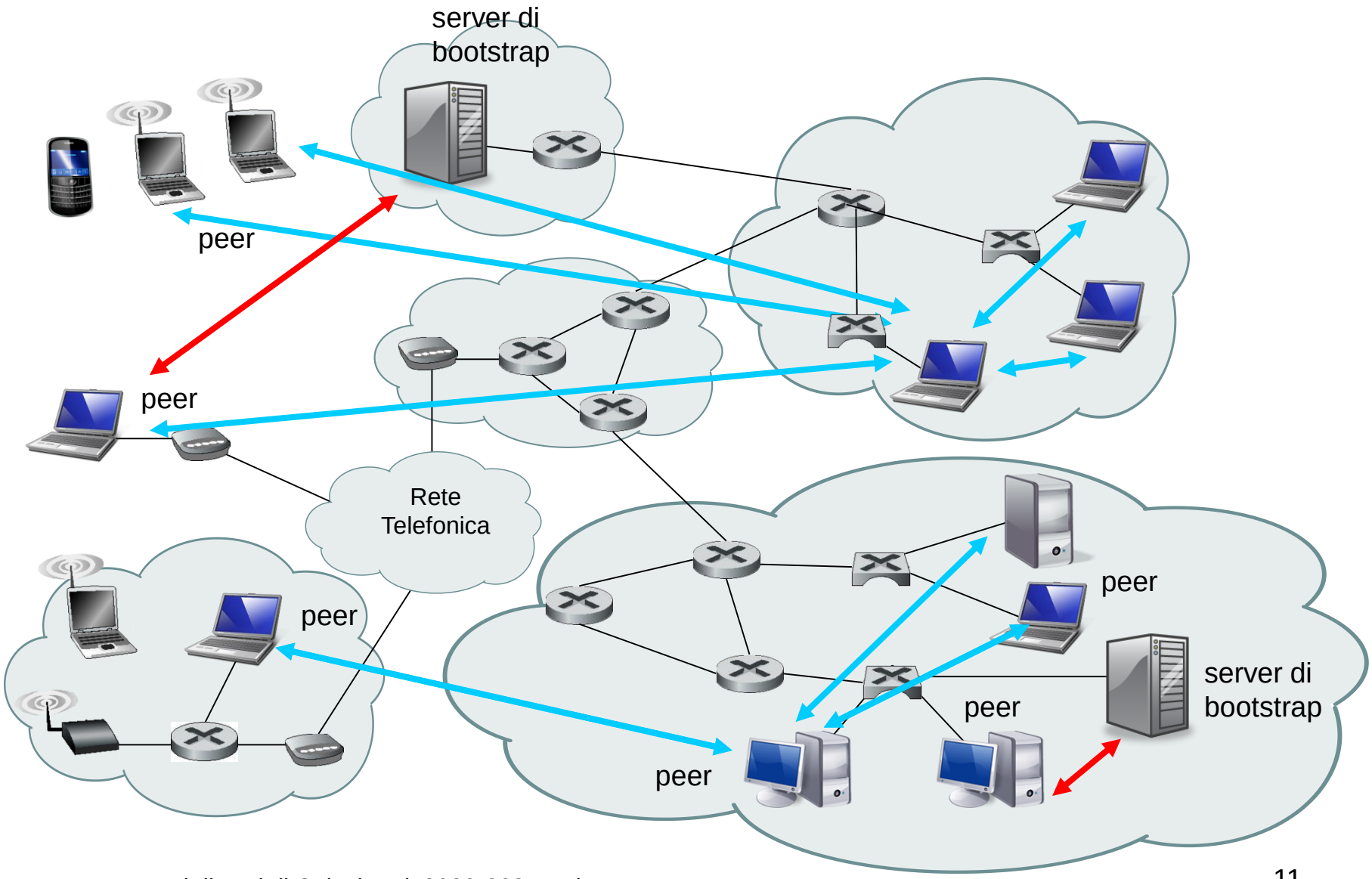
- DASH consente inoltre a un client di adattarsi alla larghezza di banda disponibile se la larghezza di banda end-to-end disponibile cambia durante la trasmissione.
- Questa funzione è particolarmente importante per gli utenti di dispositivi mobili, che spesso vedono la larghezza di banda fluttuare mentre si muovono rispetto alle stazioni base.
- Con DASH, ogni versione video è memorizzata nel server HTTP, ognuna con una diversa URL. Ogni video ha un file manifest che viene inviato al client all'inizio della trasmissione. Il file manifest è un file contenente metadati che informano, in particolare, il client sugli URL delle varie versioni e sui rispettivi bit rate del video. Il client seleziona quindi un blocco alla volta specificando un URL e la dimensione del blocco in byte in un messaggio di richiesta HTTP GET per ogni blocco.
- Durante il download dei blocchi, il client misura anche la larghezza di banda corrente ed esegue un algoritmo per determinare da quale versione del file selezionare il blocco da richiedere successivamente.

- Naturalmente, se ha bufferizzato una buona quantità di video e se la larghezza di banda di ricezione misurata è elevata, sceglierà un blocco da una versione a bitrate elevato. D’altra parte, se il client ha poco video nel buffer e la larghezza di banda ricevuta misurata è bassa, sceglierà un blocco da una versione a basso bitrate.
- Con DASH quindi il client può passare automaticamente a prelevare blocchi di video a diversi livelli di qualità.

Applicazioni P2P (da pari a pari)

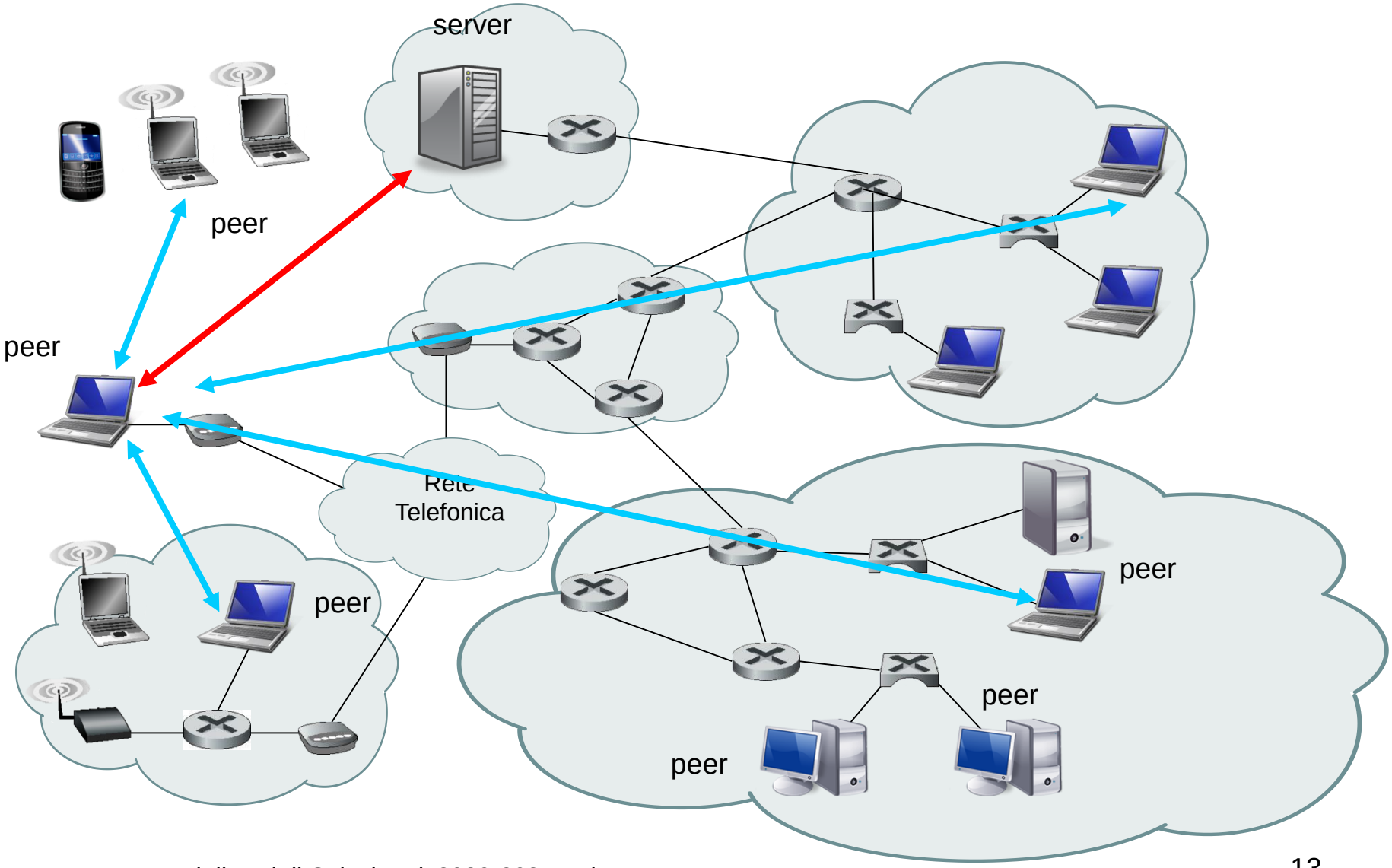
- I protocolli e le applicazioni finora descritti sono stati progettati con architettura client/server.
- Un altro modello, più complesso, per la progettazione di applicazioni di rete è il Peer to Peer (P2P).
- Questa architettura consente agli host di scambiarsi file direttamente tra loro, senza passare attraverso server intermediari.
- Gli host in questa architettura sono detti **pari**.
- In ogni pari è implementato sia il lato client sia il lato server del protocollo di trasferimento di file che generalmente è l'HTTP (a volte FTP), poiché un pari è sia un distributore che un fruitore di contenuti.
- Nel P2P, ci sono anche aspetti importanti legati alla violazione del copyright, della privacy, e della sicurezza.
- In un sistema P2P in ogni istante sono connessi un gran numero di pari, e ogni pari può avere molti file da condividere.

- Un importante servizio che un'applicazione P2P deve fornire agli utenti è la **ricerca dei contenuti**. Se un pari **P** vuole ottenere un particolare file, ad esempio un brano musicale, allora quel pari **P** deve conoscere gli indirizzi IP dei pari connessi che hanno copie del file desiderato.
- Vedremo tre architetture per ricercare i file, ciascuna delle quali è stata usata da diversi sistemi di condivisione di file P2P.
- Nella figura seguente è mostrata l'architettura P2P.


server di
bootstrap
peer
peer
Rete
Telefonica
peer
peer
peer
peer
server di
bootstrap

Database centralizzato

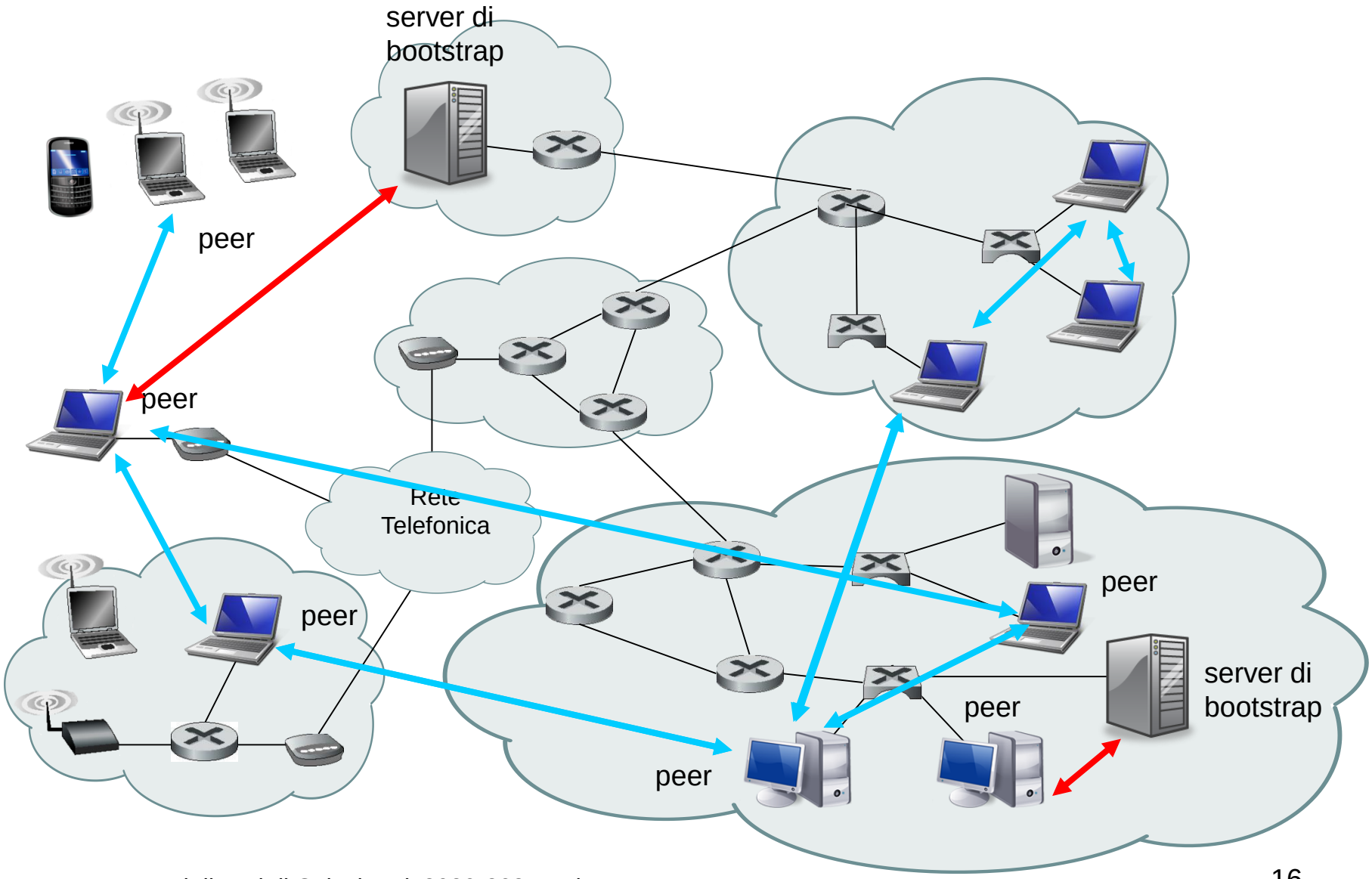
- Una delle soluzioni più semplici per ricercare i file da scaricare si ottiene mediante un server che gestisce un database centralizzato, come nel caso di Napster.
- Come è mostrato nella figura, al momento dell'avvio, l'applicazione P2P si connette a un server inviando ad esso i nomi dei file contenuti nelle sue cartelle locali condivise.
- Il server memorizza le informazioni provenienti da tutti i pari che si connettono al sistema, creando in tal modo un database centralizzato, dinamico che associa ogni nome di file con un insieme di indirizzi IP di pari che possiedono il file. Quindi, il database contiene semplicemente coppie **(chiave, valore)**. Le chiavi,ad esempio, potrebbero essere i nominativi dei contenuti come titoli di film, di brani musicali o di nomi di software e i valori sono gli indirizzi IP e il numero di porta dei pari dove i contenuti sono memorizzati. Un esempio di coppia (chiave, valore) è (Lucio Battisti – Sì viaggiare, 180.20.10.122:4010).

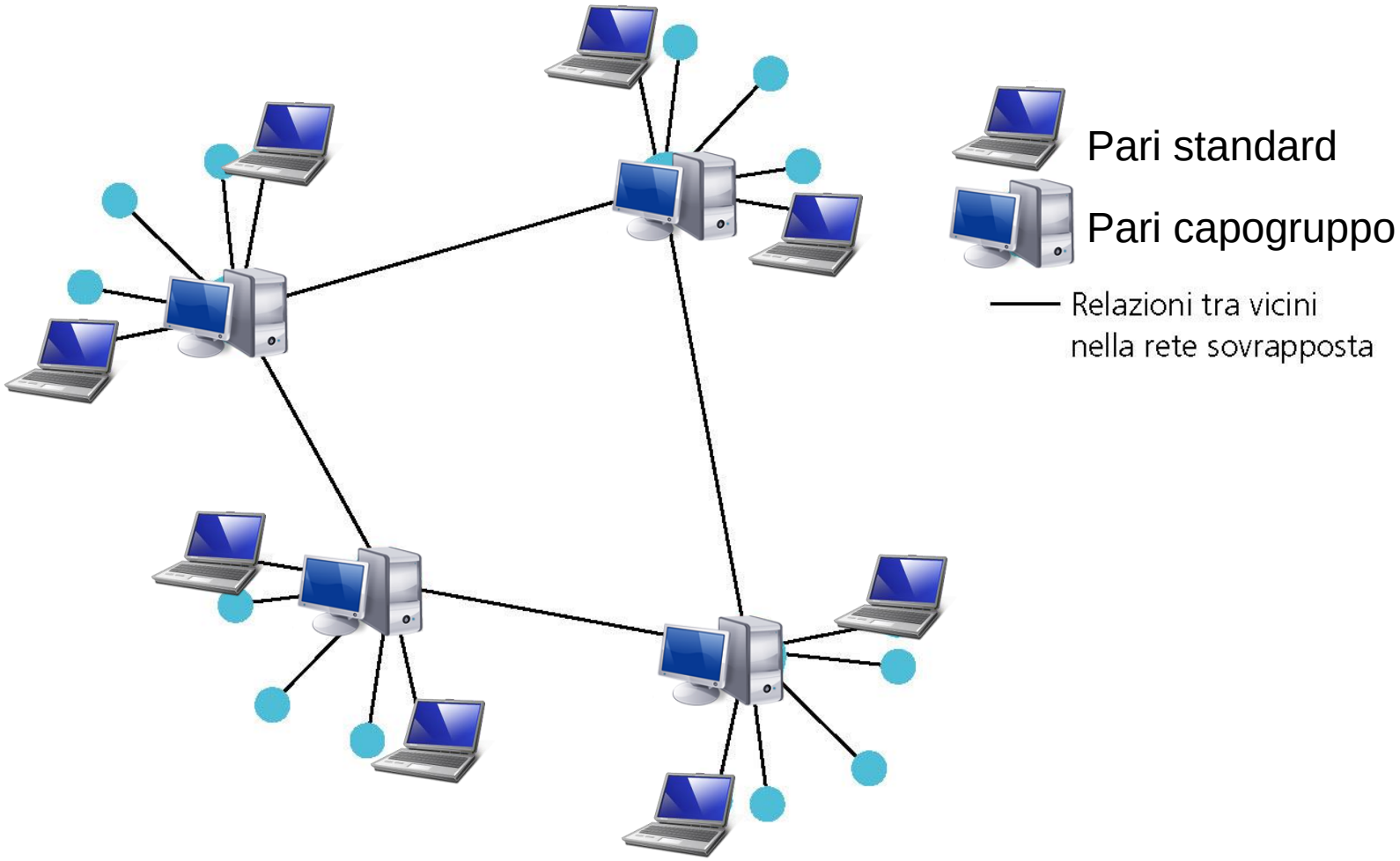

server
peer
peer
Rete
Telefonica
peer
peer
peer
peer

- Quando un pari scarica o rimuove o aggiunge un nuovo file nella cartella condivisa, informa il server in modo che esso possa aggiornare il database. Per mantenere aggiornato il database il server deve sapere quando un pari si disconnette dalla rete. Un modo per conoscere quali pari sono connessi è di inviare periodicamente messaggi ai pari per verificare se sono connessi. Se il server rileva che un pari non è più connesso, rimuove il suo indirizzo IP e tutte le sue informazioni dal database.
- Usare un database centralizzato per localizzare i file ha una serie di svantaggi:
  - ***Unico punto di guasto****.* Se il server non funziona o non è raggiungibile per qualsiasi motivo allora si blocca l'intera applicazione P2P.
  - ***Collo di bottiglia per le prestazioni****.* In un grande sistema P2P, con centinaia di migliaia di utenti connessi, un server centralizzato deve mantenere aggiornato un enorme database e deve rispondere a centinaia o migliaia di richieste al secondo, pertanto le prestazioni del servizio sono limitate da problemi di sovraccarico del server e di traffico.

Database distribuito

- Per superare i limiti di scalabilità e di prestazioni si può realizzare il database, per la ricerca dei contenuti, distribuito tra un insieme di pari detti **capogruppo**.
- Un pari quando si connette alla rete P2P si connette con qualche altro pari già appartenente alla rete.
- Pertanto, il sistema P2P deve avere uno o più **server di bootstrap** sempre accesi, che svolgono il compito di gestire e mantenere la topologia della rete P2P.
- Quando un pari si connette alla rete P2P, prima contatta un server **di bootstrap** il quale può assegnargli il ruolo di capogruppo, oppure inviargli l'indirizzo IP di uno dei capogruppo già esistenti, e quindi il pari si connette a quel capogruppo.
- Se il pari è stato nominato capogruppo, il server di bootstrap invia ad esso gli indirizzi IP di alcuni altri capigruppo.
- Il capogruppo gestisce, per tutti i pari del suo gruppo, un database che associa i file con gli indirizzi IP.

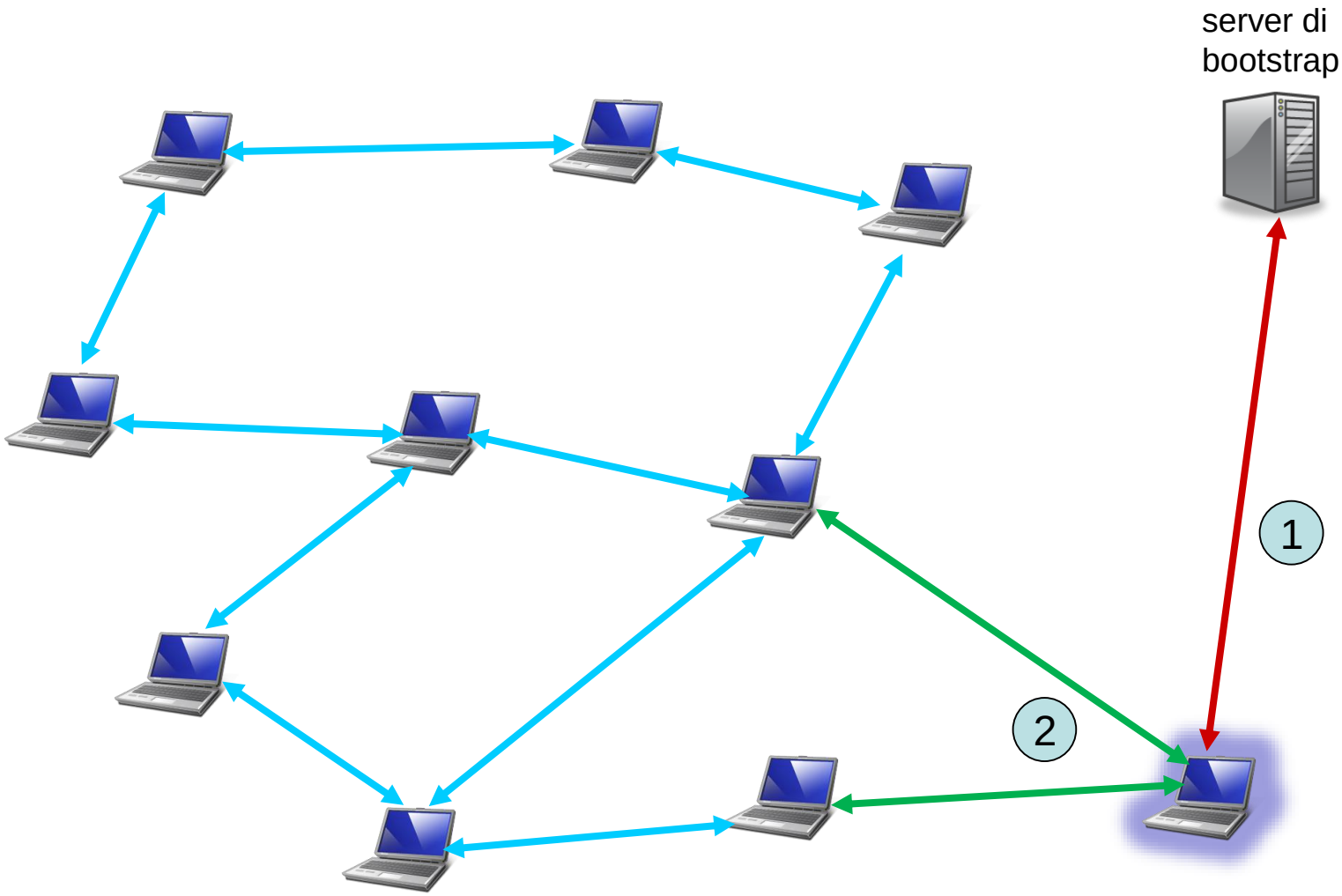

server di
bootstrap
peer
peer
Rete
Telefonica
peer
peer
peer
peer
server di
bootstrap


Pari standard
Pari capogruppo
Relazioni tra vicini
nella rete sovrapposta

- Con questa architettura, la rete è costituita da un insieme di sistemi P2P con database centralizzato, in cui ogni capogruppo ha un ruolo sia di database server che di pari.
- Pertanto, i database hanno minori dimensioni e ogni capogruppo riceverà meno richieste e il traffico sarà più distribuito. L'insieme di pari e delle loro connessioni logiche, formano una rete logica, detta **rete di copertura (o sovrapposta)**. Poiché sia i pari che i capogruppo si connettono e disconnettono continuamente, la rete di copertura cambia dinamicamente. Questa architettura richiede di realizzare un protocollo molto complesso per gestire la rete di copertura. Ad esempio, quando un capogruppo si disconnette, è necessario assegnare tutti i pari del capogruppo ad altri capigruppo.
- Nella fase di ricerca di un file, il pari invia la richiesta al proprio capogruppo il quale a sua volta la rinvia agli altri capogruppo per ottenere un elenco completo dei pari che hanno il file richiesto. In questo modo il pari richiedente ottiene molte liste di pari che hanno il file richiesto.

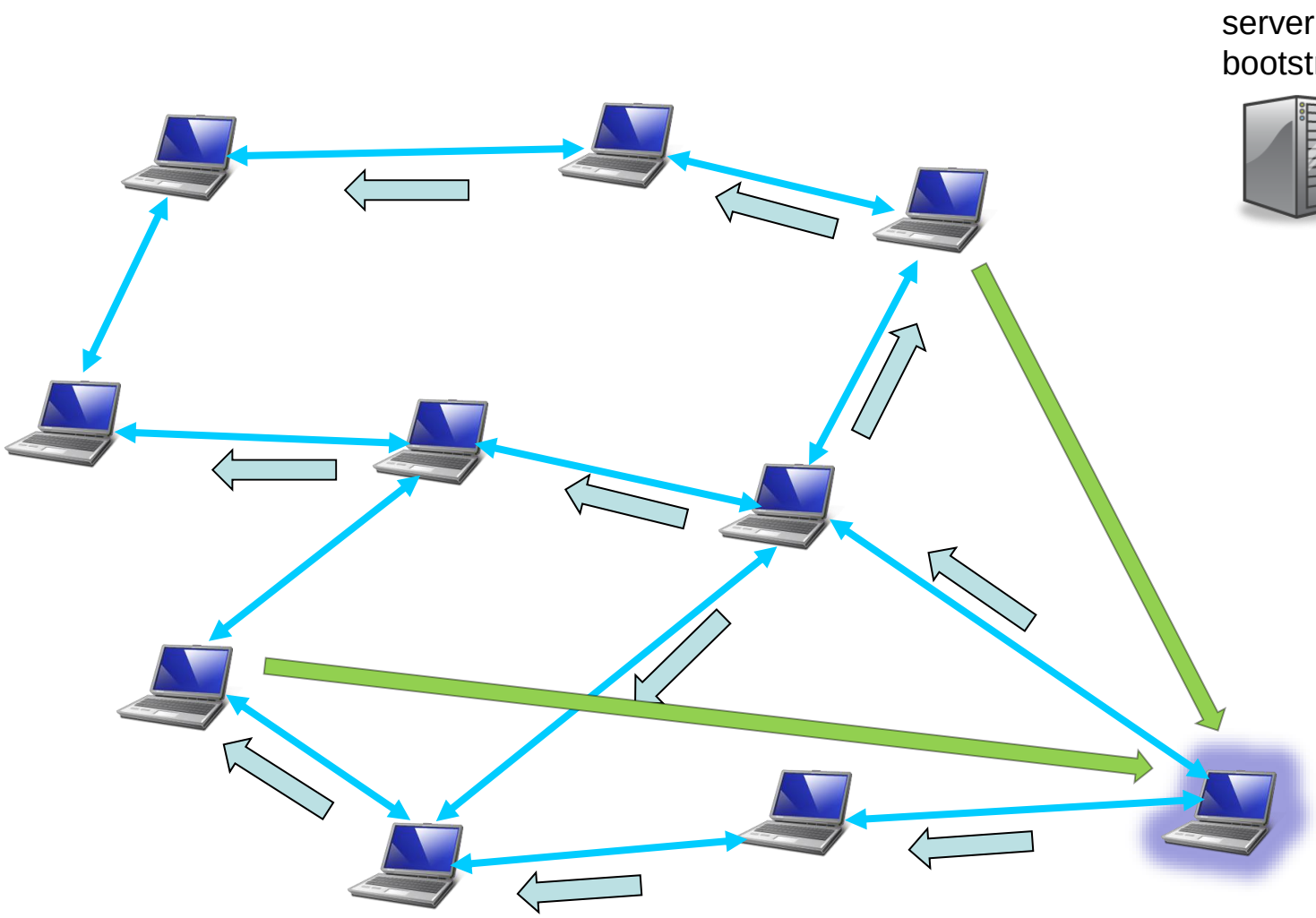
Inondazione di richieste (query flooding)

- L’applicazione P2P open source Gnutella usa un approccio completamente distribuito per realizzare la ricerca dei file.
- In Gnutella, non esiste una struttura gerarchica con pari e capigruppo ma i pari si uniscono in una rete di copertura senza gerarchia.
- Anche Gnutella utilizza nodi di bootstrap sempre attivi.
- Un pari si unisce alla rete P2P, **connettendosi a un server di bootstrap** che comunica al pari gli indirizzi IP di un certo numero di pari appartenenti alla rete di copertura.
- Ogni pari comunica solo con i pari suoi vicini. Anche per la realizzazione di questa architettura è richiesto un protocollo complesso per mantenere aggiornata la rete sovrapposta, poiché i pari si connettono e disconnettono continuamente.



Unione di un pari alle rete P2P Gnutella

- Per la ricerca dei file, Gnutella usa la tecnica detta **inondazione di richieste *(query flooding).***
- Con tale tecnica, quando un pari ricerca un file, invia una richiesta a tutti i suoi vicini, ciascuno dei quali rinvia la richiesta a tutti i pari suoi vicini, meno che al richiedente. In questo modo, la richiesta arriva ad ogni pari appartenente alla rete sovrapposta. Se un pari che riceve la richiesta ha una copia del file desiderato, invia una risposta al pari richiedente, indicando che possiede una copia del file.
- La tecnica del **flooding** crea una elevato traffico di richieste, per limitare il quale, si stabilisce un limite massimo di livello di propagazione delle richieste. Ciò si realizza con un campo ***contatore di nodo*** presente nel messaggio di richiesta, che inizialmente è impostato ad un valore predefinito, ad esempio 9. Ogni volta che il messaggio di richiesta arriva ad un pari, il campo contatore viene decrementato di uno prima che il messaggio di richiesta sia rinviato al suoi vicini. Quando un pari riceve una richiesta con il campo contatore uguale a zero, non la inoltra.



Ricerca di contenuti nella rete P2P Gnutella

Tabelle hash distribuite (DHT)

- Molti sistemi P2P attuali hanno un'architettura distribuita basata sulla **tabella hash distribuita (DHT, distributed hash table).**
- In questo modello P2P ogni pari gestisce solo un piccolo numero di coppie (chiave, valore).
- Ogni pari può interrogare il database distribuito con una specifica chiave; il database distribuito quindi troverà i pari che hanno le coppie (chiave, valore) corrispondenti e le invierà al pari che le ha richieste.
- Inoltre, ogni pari potrà inserire nel database nuove coppie (chiave, valore).

- Con DHT ogni pari è identificato da un numero intero contenuto nell'intervallo [0, 2^N - 1] per un N fissato.
- Anche ogni chiave è rappresentata da un intero nello stesso intervallo.
- In pratica, un utente usa chiavi di ricerca testuali, come ad esempio il titolo di un libro, di un brano musicale, non numeri interi. Per ottenete chiavi di numeri interi (o tipo string) a partire da chiavi alfanumeriche si usa una **funzione hash**

  KEY_HASH = F_HASH(KEY_STRING)

  che crei la corrispondenza tra le chiavi testuali e gli interi nell'intervallo [O, 2^N - 1] .
- Una funzione hash è una funzione molti a uno, per cui due input diversi possono avere lo stesso output (stesso numero

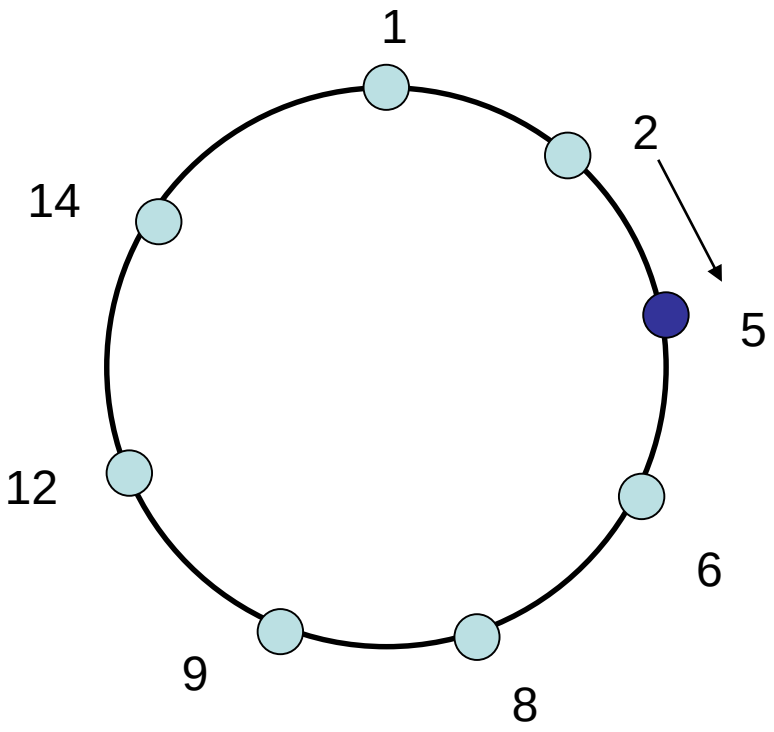
intero), ma con probabilità estremamente piccola. Esempi di funzioni hash sono MD5 e SHA (Secure Hash Algorithm).

- Quindi, per esempio, se un utente vuole memorizzare o ricercare il brano "Lucio Battisti – Sì viaggiare", la chiave usata nella DHT sarà l'intero corrispondente alla funzione hash di tale titolo (o del contenuto del file).
- Consideriamo ora il problema di memorizzare le coppie (chiave, valore) nella DHT. Il problema principale è stabilire una regola per assegnare la gestione delle coppie (chiave, valore) ai pari. Poiché ogni pari è identificato da un numero intero e ogni chiave è rappresentata da un intero nello stesso intervallo, la soluzione più semplice è di assegnare la gestione di ogni coppia (chiave, valore) al pari il cui identificatore ha il valore più vicino alla chiave.
- Per chiarire questo schema consideriamo un piccolo esempio. Assumiamo che sia N = 4 in modo che tutti gli identificatori dei pari e delle chiavi siano compresi nell'intervallo [0, 15] e che ci siano 8 pari nel sistema con identificatori 1, 2, 5, 6, 8, 9, 12, 14.

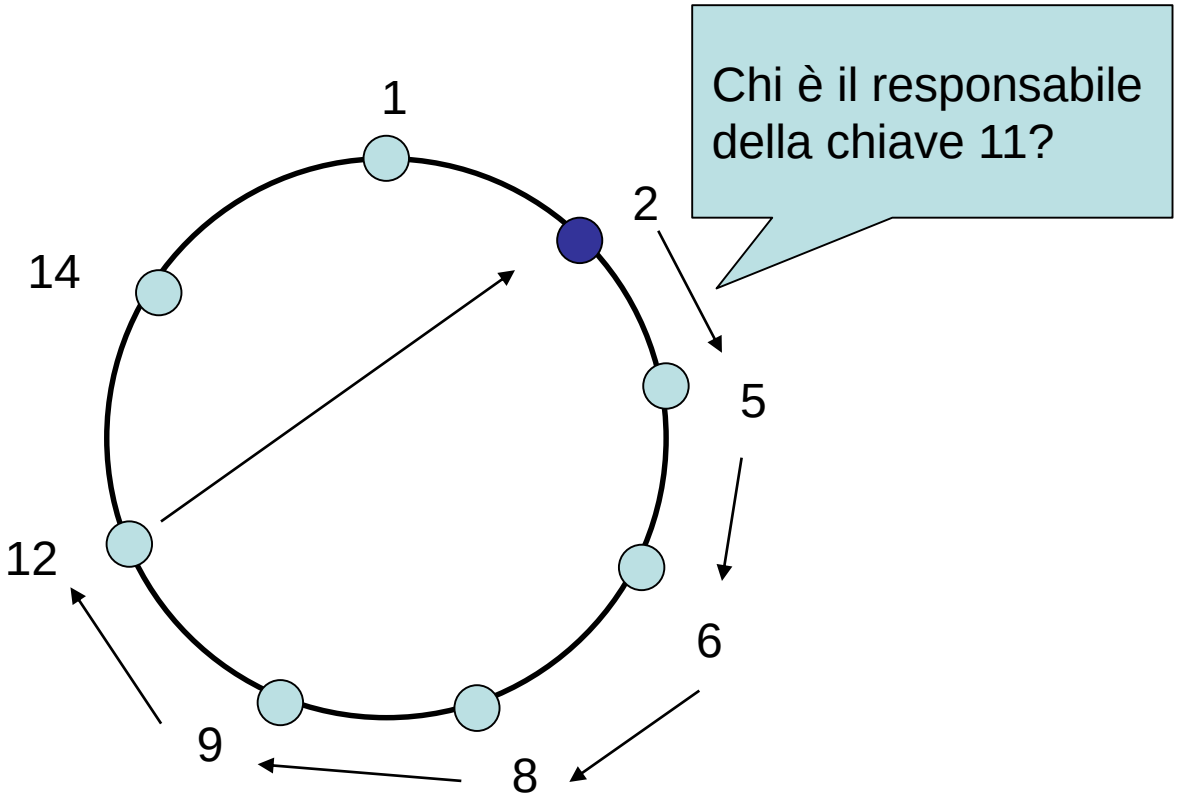
- Quindi, supponiamo di voler memorizzare la coppia (11 , "Lucio Battisti – Sì viaggiare") in uno degli 8 pari.
- Usando la tecnica del "più vicino", poiché il pari 12 è il valore più vicino alla chiave 11, memorizziamo la coppia (11, "Lucio Battisti – Sì viaggiare ") nel pari 12.
- Supponiamo ora che un pari P voglia inserire una coppia (chiave, valore) nella DHT. Per farlo deve prima determinare il pari il cui identificatore ha valore più vicino alla chiave, poi invia ad esso un messaggio di *richiesta di memorizzazione* per la coppia (chiave, valore).
- A questo punto, il problema da risolvere è identificare il pari più vicino alla chiave. La semplice soluzione che prevede che ogni pari possieda tutte le associazioni tra identificatore del pari e corrispondente indirizzo IP sarebbe impraticabile per sistemi su larga scala con centinaia di migliaia o milioni di pari.

DHT circolare

- Per risolvere il problema di scalabilità organizziamo i pari in una lista circolare. In questo schema circolare ogni pari tiene traccia solo dei suoi vicini, precedente e successivo. Per esempio, il pari 5 conosce l'indirizzo IP e l'identificatore dei pari 2 e 6, ma non conosce gli altri pari che possono esserci nella DHT.



- In relazione alla figura, supponiamo ora che il pari 2 voglia conoscere in quale pari nella DHT sia memorizzata la chiave 11.
- Usando la copertura circolare, il pari 2, per identificare il pari responsabile della chiave 11, invia un messaggio a uno dei due suoi vicini, ad esempio al suo successivo, il pari 5.

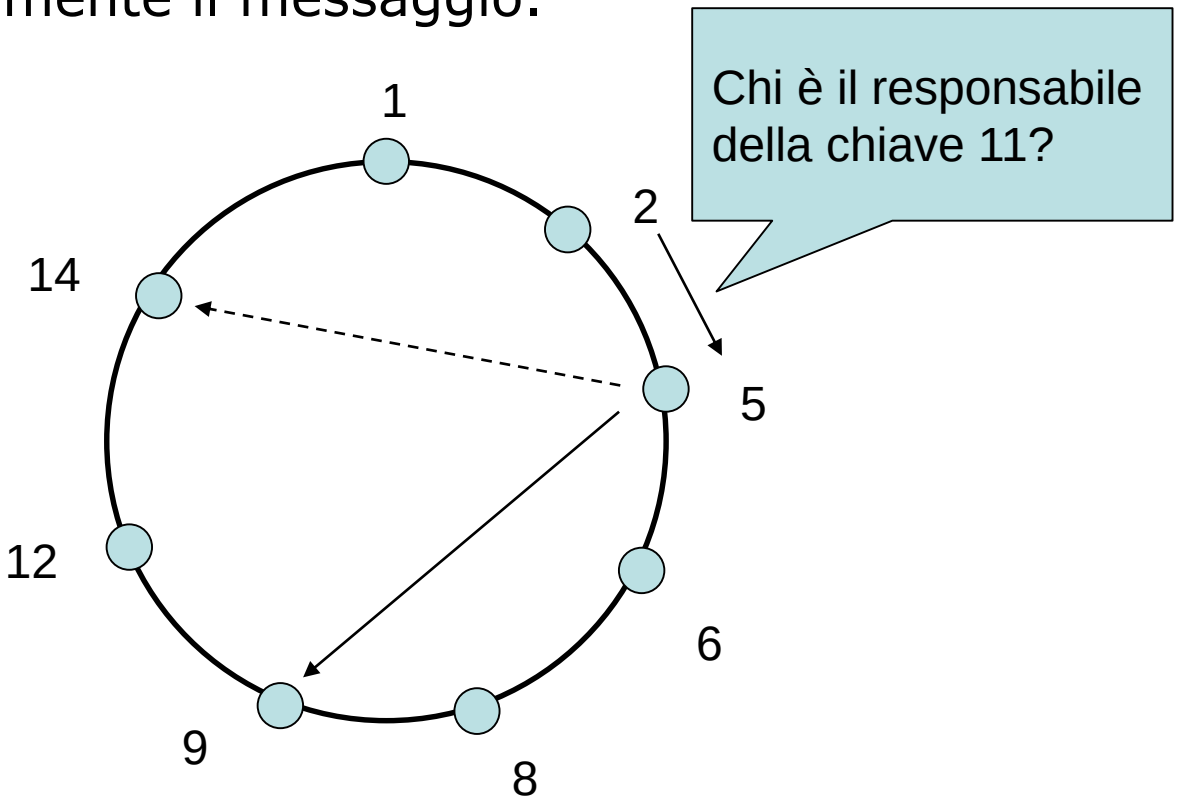


- Ogni volta che un pari riceve questo messaggio, poiché conosce l'identificatore del suo pari successivo e del suo precedente, può determinare se sia responsabile (cioè se è il più vicino) della chiave in questione. Se il pari non è responsabile della chiave, rinvia il messaggio al pari successivo. Quindi, per esempio, quando il pari 5 riceve il messaggio di richiesta della chiave 11, determina di non essere responsabile di tale chiave, perché il suo pari successivo nella DHT è più vicino alla chiave, e passa il messaggio al pari 6.
- Questo processo continua finché il messaggio arriva al pari 12 che determina di essere il pari più vicino alla chiave 11. A questo punto, il pari 12 può rispedire il messaggio al pari richiedente, il pari 2, indicandogli di essere responsabile della chiave 11.

- La DHT circolare riduce notevolmente la quantità di informazioni che ogni pari deve gestire. In particolare, ogni pari deve conoscere l'identità di soli due pari, il suo pari successivo e precedente.
- Tuttavia, questa soluzione richiede che, per trovare il nodo responsabile di una chiave, nel caso più sfortunato, tutti gli N nodi della DHT debbano rinviare il messaggio. In media vengono inviati N/2 messaggi.
- Così, nella progettazione di una DHT è necessario trovare un compromesso tra il numero di vicini di cui ogni pari deve conoscere e il numero di messaggi che la DHT deve trasmettere per risolvere una singola interrogazione.

- Da una parte, se ogni pari tiene traccia di tutti gli altri pari (mesh overlay), allora viene inviato solo un messaggio per interrogazione, ma ogni pari deve conoscere tutti gli altri N-1 pari. D’altra parte, in una DHT circolare, ogni pari conosce solo due pari, ma vengono inviati in media N/2 messaggi per ogni interrogazione.
- Un compromesso si può ottenere tenendo come base la copertura circolare, e aumentando il numero di vicini, in modo che ogni pari non tenga traccia solo del suo pari successivo e precedente, ma anche di un numero, relativamente piccolo, di altri pari vicini.
- Un esempio di DHT circolare di questo tipo con più vicini è mostrato nella figura seguente. Nella figura sono messi in evidenza solo i vicini del pari 5.

- Quando un pari riceve un messaggio di interrogazione per una chiave, lo inoltra al vicino (il successivo o uno degli altri vicini) che ha identificatore più vicino alla chiave.
- Quindi, nella figura, quando il pari 5 riceve il messaggio di interrogazione della chiave 11, determina che il pari più vicino alla chiave (tra i suoi vicini) è il vicino 9 e quindi gli inoltra direttamente il messaggio.



- Chiaramente, l’aggiunta di ulteriori vicini può ridurre in modo significativo il numero di messaggi usati per elaborare un’interrogazione.
- Un importante punto nella progettazione DHT è calcolare il numero dei vicini che ciascun pari dovrebbe avere.
- Molti studi hanno dimostrato che la DHT può essere progettata in modo che sia il numero di vicini per pari sia il numero di messaggi per interrogazione sia O(log NP), dove NP è il numero di pari.
- Questa architettura raggiunge un compromesso soddisfacente tra le due soluzioni estreme di topologia di copertura circolare e mesh overlay.

Connessione e disconnessione dei pari

- Nei sistemi P2P i pari si connettono e disconnettono continuamente. Quindi nella progettazione delle DHT è necessario mantenere continuamente aggiornata la rete di copertura.

BitTorrent

- BitTorrent è un protocollo P2P per la distribuzione di file molto diffuso. In BitTorrent, l'insieme di tutti i pari che partecipano alla distribuzione di un determinato file è detto **torrent** *(torrente).*
- I pari appartenenti a un torrent si scambiano parti *(chunk)* del file tra loro. La dimensione tipica dei chunk è 256 kbyte.
- Quando un pari si unisce a un torrent per la prima volta, non ha parti del file. Col passare del tempo, memorizza sempre più parti che, mentre scarica, invia ad altri pari.
- Un pari può disconnettersi dal torrent in qualsiasi momento con solo alcune parti del file e riconnettersi al torrent successivamente.
- Ciascun torrent è gestito da un server detto **tracker**.
- Quando un pari entra a far parte di un torrent, si registra presso il tracker e periodicamente invia messaggi al tracker per comunicare la propria appartenenza al torrent.

- In questo modo, il tracker crea una mappa dei pari che appartengono al torrent. In un dato istante, un torrent può avere migliaia di pari.
- Quando un nuovo pari **P**, entra in un torrent, il tracker seleziona in modo casuale un sottoinsieme di pari (ad esempio 40) dall'insieme dei pari che appartengono a quel torrent, e invia l'indirizzo IP di questi 40 pari al nuovo pari. Ottenuta la lista dei pari, il nuovo pari **P** cerca di stabilire connessioni TCP con tutti i pari della lista. Indichiamo con ***"pari vicini"*** i pari con i quali il nuovo pari **P** riesce a stabilire una connessione TCP.
- Col passare del tempo, alcuni pari vicini possono disconnettersi dal torrent, mentre altri possono cercare di stabilire una connessione TCP con **P**. Quindi per ciascun pari, i vicini cambiano continuamente.
- In un certo istante, ogni pari avrà un sottoinsieme delle parti di un file.

- Periodicamente, il nuovo pari chiederà a ciascuno dei suoi vicini la lista delle parti del file in loro possesso.
- Tramite questa conoscenza, il pari **P** invierà richieste per le parti del file che ancora non possiede. Pertanto, in un dato istante, il pari **P** avrà alcune parti del file e saprà quali parti hanno i suoi vicini.
- Ne richiedere le parti ai suoi vicini il pari **P** adotta la tecnica «il più raro per primo» che consiste nel richieder tra le parti che ancora non ha, quelle che sono più rare; cioè le parti che sono meno duplicate tra i suoi vicini, e richiederle per prime. In questo modo, le parti più rare sono redistribuite più velocemente, cercando di rendere uguale il numero di copie di ciascuna parte nel torrent.

Per determinare a quali richieste dei suoi vicini **P** debba rispondere, BitTorrent usa un algoritmo di scambio che stabilisce che **P** assegni una priorità più alta ai vicini che, in un dato istante, inviano dati alla velocità più alta. Pe fare questo, **P** misura continuamente la velocità di trasferimento per ciascuno dei suoi vicini, e individua un numero di pari (ad esempio 4) da cui riceve dati alla velocità più alta. **P** poi contraccambia inviando le parti del file a quegli stessi 4 pari. Ogni 10 secondi ricalcola le velocità ed eventualmente modifica l'insieme dei quattro pari.

Un punto importante è che ogni 30 secondi sceglie casualmente un vicino in più e invia ad esso le parti. Indichiamo con Q il nuovo pari scelto casualmente. Dato che P sta inviando dati a Q, potrebbe diventare uno dei quattro "uploader" principali di Q, nel qual caso Q inizierebbe a inviare dati a P. Se la velocità alla quale Q manda i dati a P è abbastanza alta, Q, a sua volta, potrebbe diventare uno dei quattro "uploader" principali di P.

In breve, ogni 30 secondi P sceglie casualmente un nuovo vicino e inizia a scambiare parti con esso. Se lo scambio di dati è soddisfacente, ciascun pari metterà l'altro nella propria lista dei quattro "uploader" principali e continueranno a scambiarsi parti tra loro fino a che uno non trova un partner migliore. La selezione casuale dei vicini consente anche a nuovi pari di avere le parti del file, in modo che abbiano qualcosa da scambiare. Tutti gli altri pari, a parte quei cinque (i quattro pari "top" e un pari di prova), non ricevono alcune parti da P.

- Nella condivisione di file P2P, un problema importante è il free-riding (sfruttamento), nel quale un pari scarica file dal sistema di condivisione, senza inviarne. L'algoritmo di scambio di BitTorrent elimina virtualmente il problema del free-riding, in quanto, se P vuole scaricare dati da Q a una velocità soddisfacente per un periodo di tempo esteso, deve nello stesso tempo inviare dati a Q a una velocità decente.