RETI DI CALCOLATORI
Autunno 2018
docente: Laura Ricci
Lesson 14:
LIVELLO APPLICAZIONE:
APPLICAZIONI P2P, BITTORRENT, NAT

slides credit:
Stefano Ferretti, Arnaud Legout (INRIA), Moreno Marzolla (Unibo)

26/11/2018

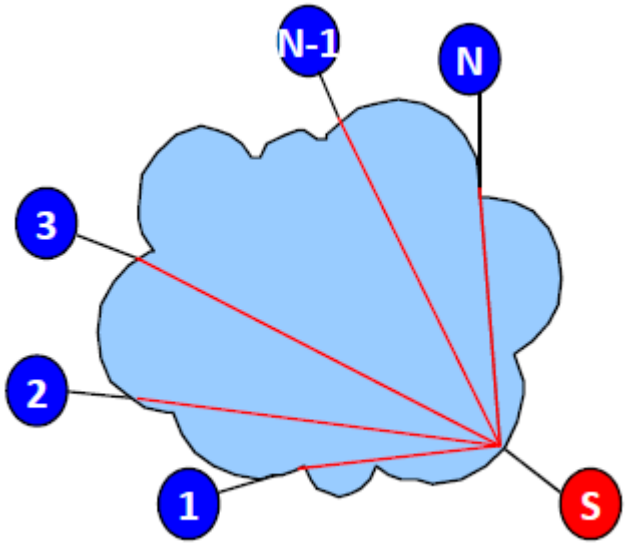
MATERIALE DIDATTICO

- capitolo 2.4
  - paragrafo 2.4.1, 2.4.2
- paragrafo 4.2.2 (NAT)

IL PARADIGMA CLIENT SERVER

- in esecuzione sugli end hosts
- comportamento on/off
- consumatore di servizi
- effettua richieste
- non comunicano direttamente tra loro
- deve conoscere indirizzo IP server

- in esecuzione sugli end host
- sempre attivo
- fornisce servizi
- riceve richieste
- soddisfa le richieste del client
- necessario un IP noto (o nome simbolico)

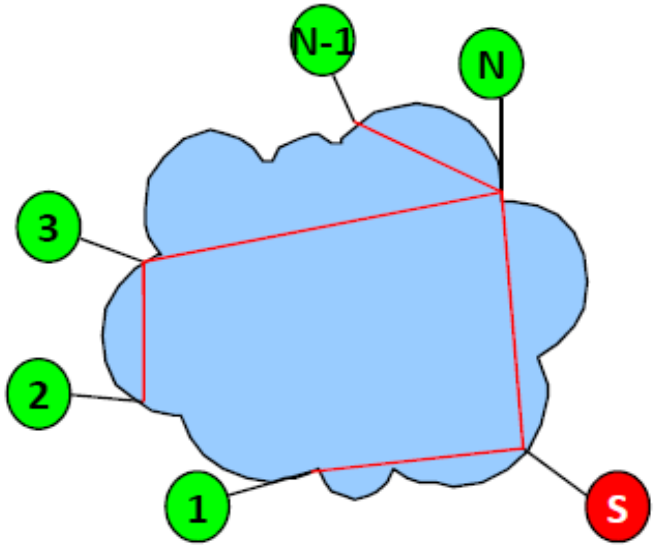


IL PARADIGMA PEER TO PEER

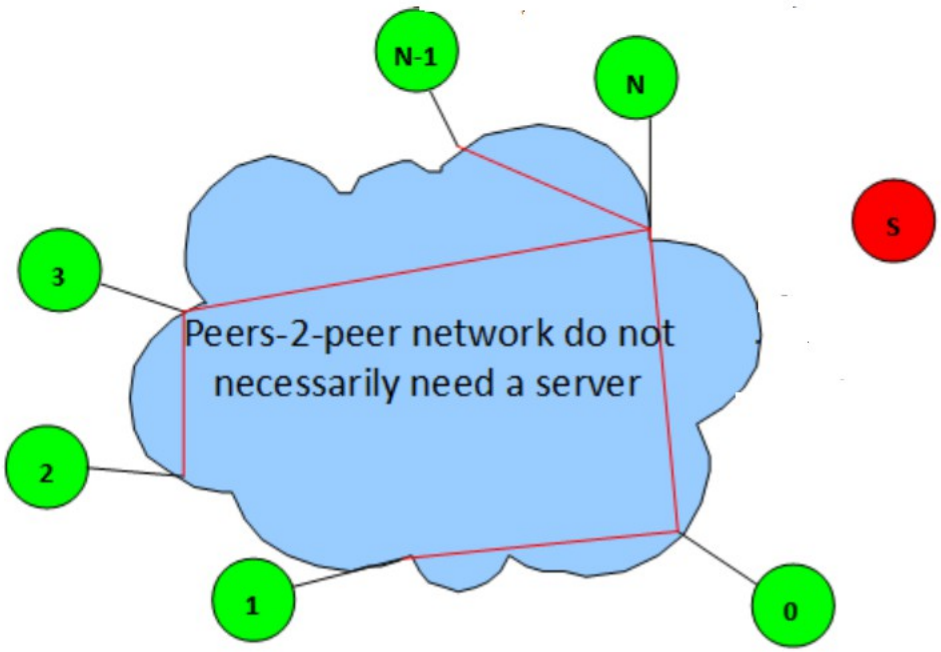
Peer

In esecuzione sugli end-host:

- comportamento on/off, **churn**
- deve unirsi alla rete
  - bootstrap
- deve poi scoprire e connettersi ad altri peer
- service providers and consumers
- comunicano direttamente tra loro
- necessario definire regole di comunicazione e cooperazione
  - per prevenire il free riding
  - incentivare la partecipazione ... e la reciprocità



IL PARADIGMA PEER TO PEER



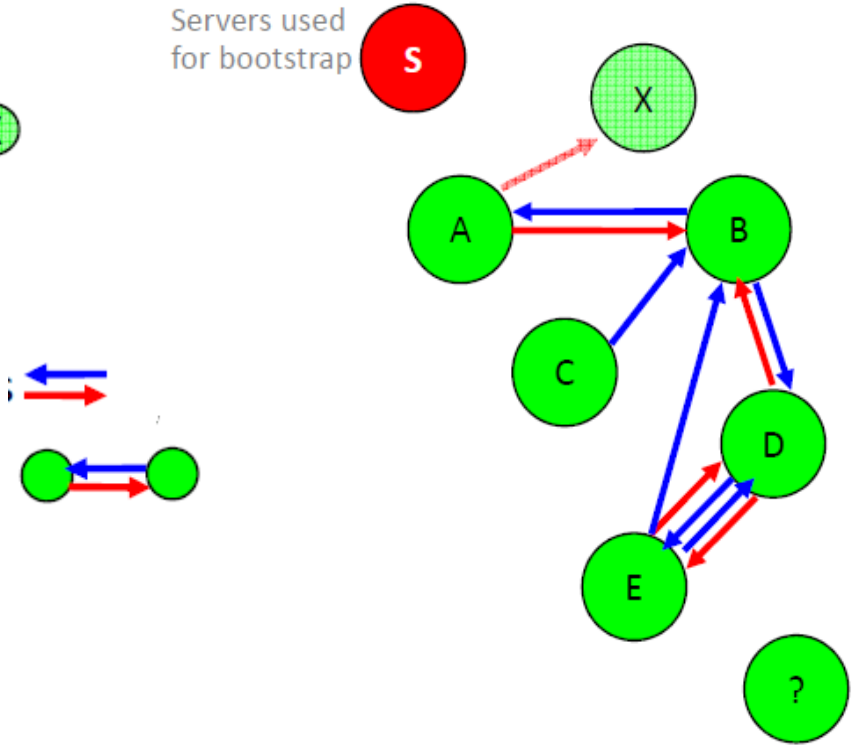
Notare che:

- un server è comunque necessario per la fase di bootstrap
- ma i server non sono utilizzati per la condivisione di risorse.

IL PARADIGMA PEER TO PEER

Peer

- in esecuzione sugli end-hosts
- comportamento on/off , churn
- Join
- deve scoprire altri peer
- service providers and consumers
- comunicazione diretta tra peers
- necessario definire un protocollo di comunicazione
- prevenire il free riding
- incentivare partecipazione e reciprocità



PEER TO PEER: UNA DEFINIZIONE GENERALE

- Definizione 1:

  Un sistema peer to peer è un insieme di entità autonome (peer) capaci di auto organizzarsi e di condividere un insieme di risorse distribuite in una rete di computer. Il sistema sfrutta le risorse per fornire il servizio in un modo completamente o parzialmente decentralizzato

- Risorse Condivise:
  - Read Only Information (Files)
  - Read/Write storage space (Distributed File System)
  - Computing power
  - Bandwidth

PEER TO PEER: UNA DEFINIZIONE GENERALE

Definizione 2:

Un sistema P2P è un sistema distribuito definito da un insieme di nodi interconnessi capaci di auto organizzarsi e di costruire topologie diverse con lo scopo di condividere risorse come cicli di CPU, memoria, banda. Il sistema è capace di adattarsi al churn continuo dei nodi, mantenendo la connettività della rete e prestazioni ragionevoli senza una entità centralizzata (come un server)

RESOURCE SHARING

- P2P: relativo a dare e ricevere da una comunità. Ogni peer condivide un insieme di risorse ed ottiene, in cambio, un insieme di risorse
  - lo scenario più comune; condividere musica (file audio) ed ottenere in cambio musica (Napster, Gnutella, …)
  - un peer si comporta sia come client che come server (funzionalità simmetrica= Servent)
- Ma un peer può decidere di offrire risorse gratis, ad esempio per partecipare ad un progetto
  - ricerca di vita extra-terrestre
  - ricerca di terapie per il cancro
  - contribuire al mantenimento di un distributed ledger
- Le risorse condivise sono "ai bordi" di Internet, sono direttamente condivise dai peer e non ci sono "special purpose nodes"per la loro gestione.

RESOURCE SHARING

- La connessione dei peer è transiente: connessioni e disconnessioni dalla rete sono frequenti

- Le risorse offerte dai peer dono aggiunte e rimosse dinamicamente

- Ogni peer è associato ad un diverso indirizzo IP address per ogni connessione al sistema

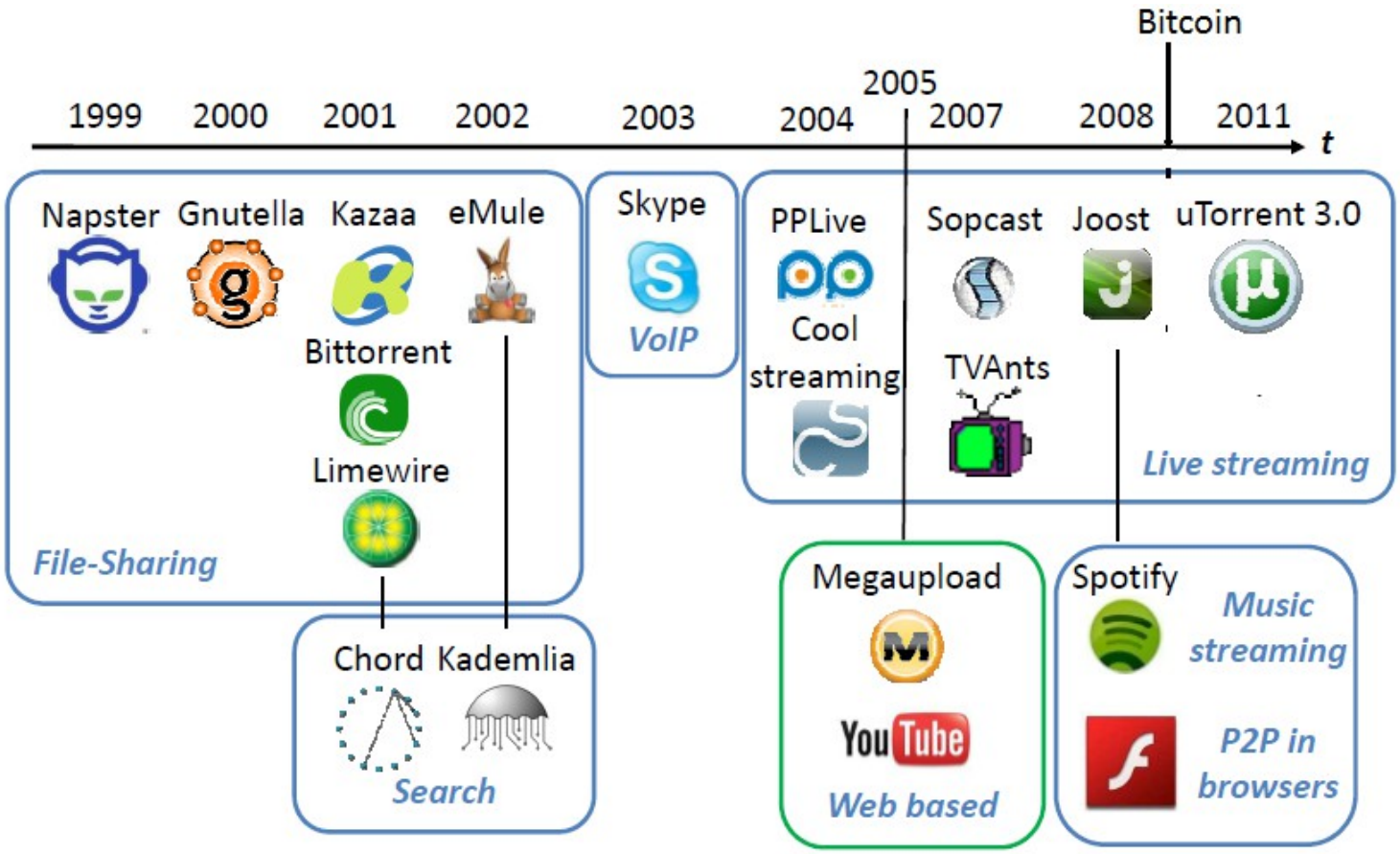
$\Rightarrow$

non è possibile individuare una risorsa tramite un IP statico IP

$\Rightarrow$

definizione di nuovi meccanismi di indirizzamento a livello apprlicazione, non a livello IP.

SISTEMI P2P: EVOLUZIONE



IL FILE SHARING: LA KILLER APPLICATION

In una applicazione di file sharing:

- un utente A ha scaricato un applicativo P2P sul suo notebook
- si connette più volte ad Internet: ogni volta ottiene un nuovo indirizzo IP, uno per ogni nuova connessione
- l'utente memorizza i file che intende condividere in una directory e descrive il file con un insieme di parole chiave (per una canzone: titolo, autore, data di pubblicazione,...)
- U è interessato a trovare una canzone ed invia una query al sistema
- il client ricerca la canzone nella rete P2P e mostra all'utente quali altri peer possiedono la canzone richiesta
- U sceglie un peer P, tra questi (secondo qualche criterio)
- il file è copiato dal notebook di P a quello di U, direttamente
- mentre U sta effettuando il download, gli altri utenti possono scaricare parti del file che U ha già scaricato e metterle in una directory comune.

CONTENT DISTRIBUTION NETWORKS

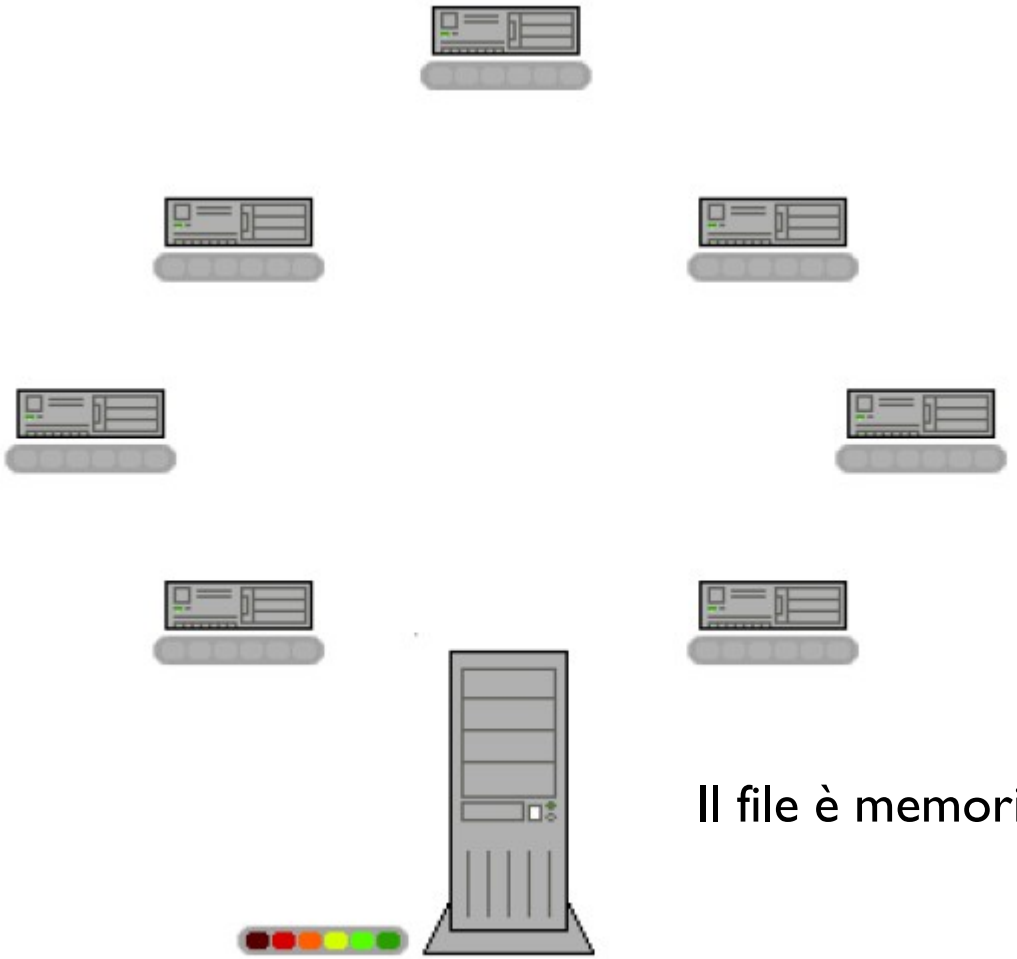
- **Content Distribution/Delivery Network (CDN)**:
  - una applicazione legata alla condivsione di risorse
  - insieme di host che cooperano per istribuire un grosso insieme di dati agli end users

- Il fenomeno del flash crowd:
  - un web server acceduto da una enorme quantità di persone in un piccolo intervallo di tempo
  - il server deve gestire, in questo periodo di tempo, una grossa mole di traffico inaspettato

- La soluzione delle CDN:
  - replicazione di dati e/o servizi su diversi mirror servers
  - scopo: ottimizzare l'uso della banda

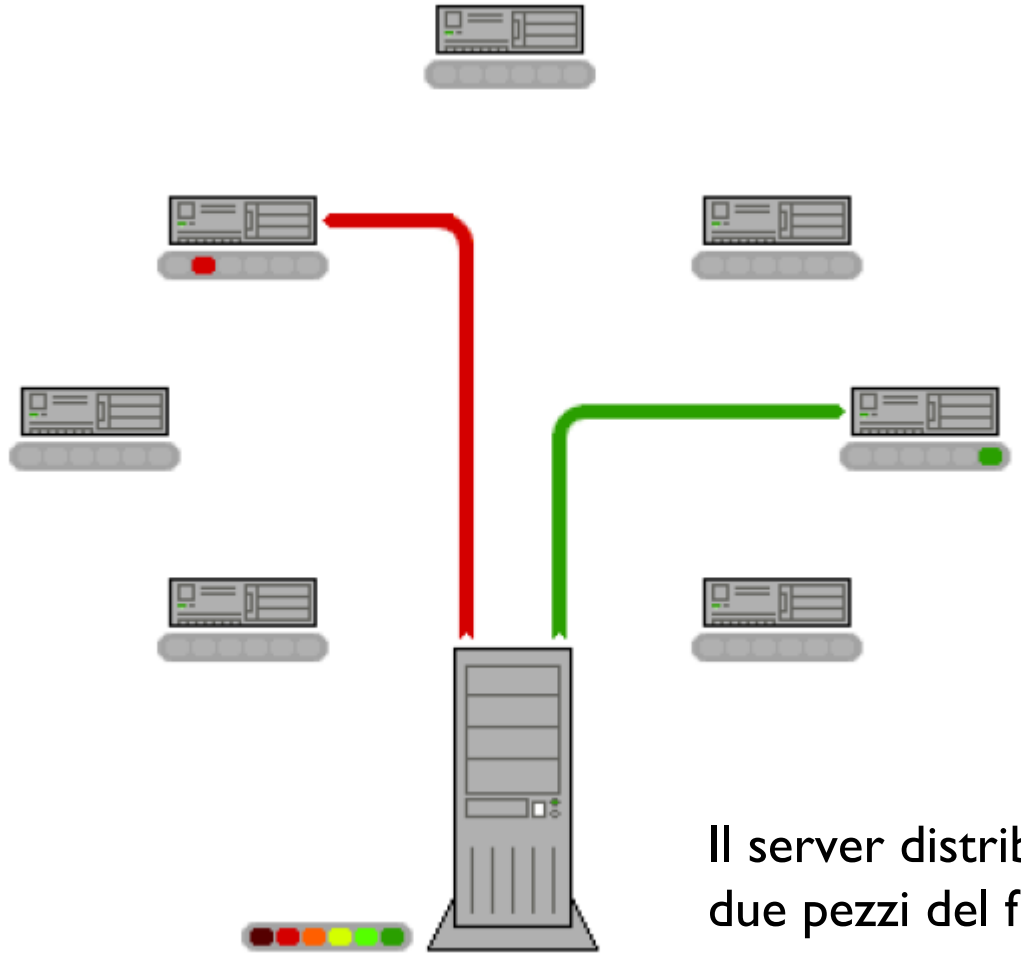
- Sistemi che adottano del tutto o in parte il paradigma P2P:
  - Commerciali: Akami, AppStream, Globix, Bittorrent
  - Accademici: Coral (P2P), Globule

CONTENT DISTRIBUTION NETWORKS



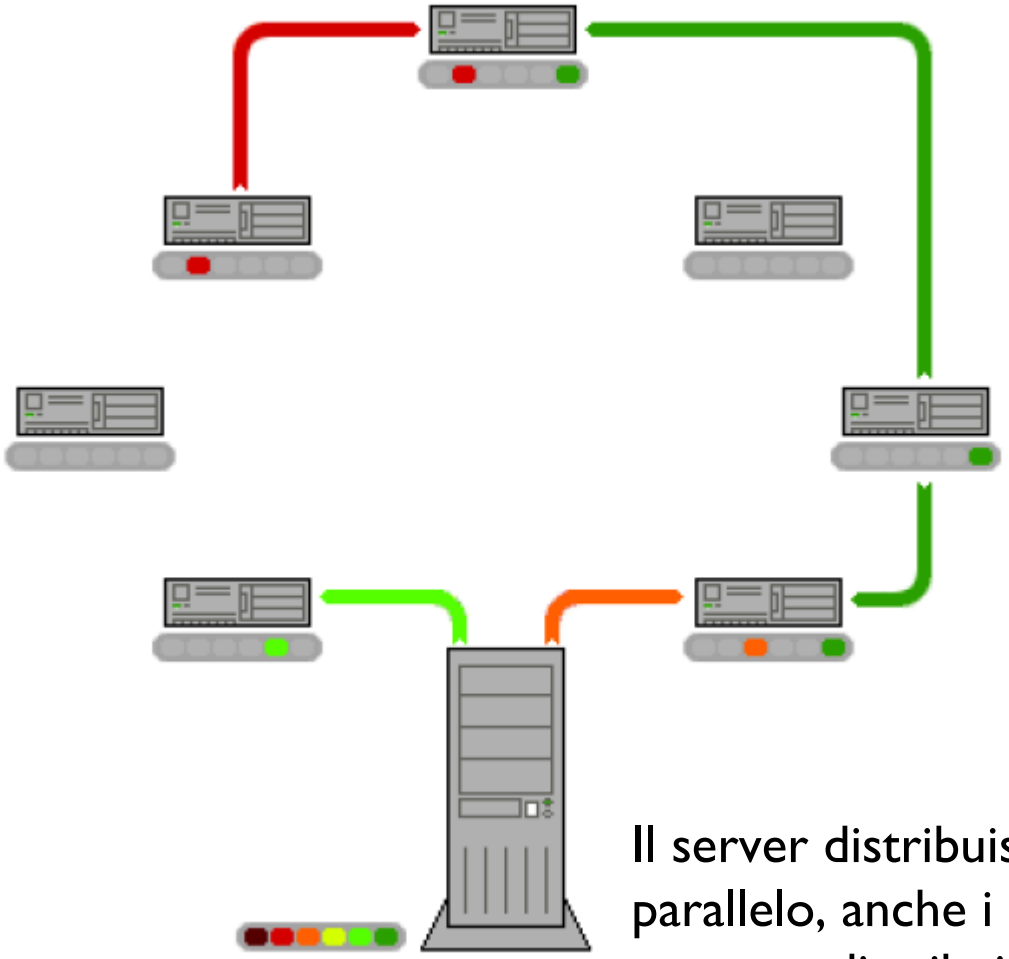
Il file è memorizzato sul server

CONTENT DISTRIBUTION NETWORKS



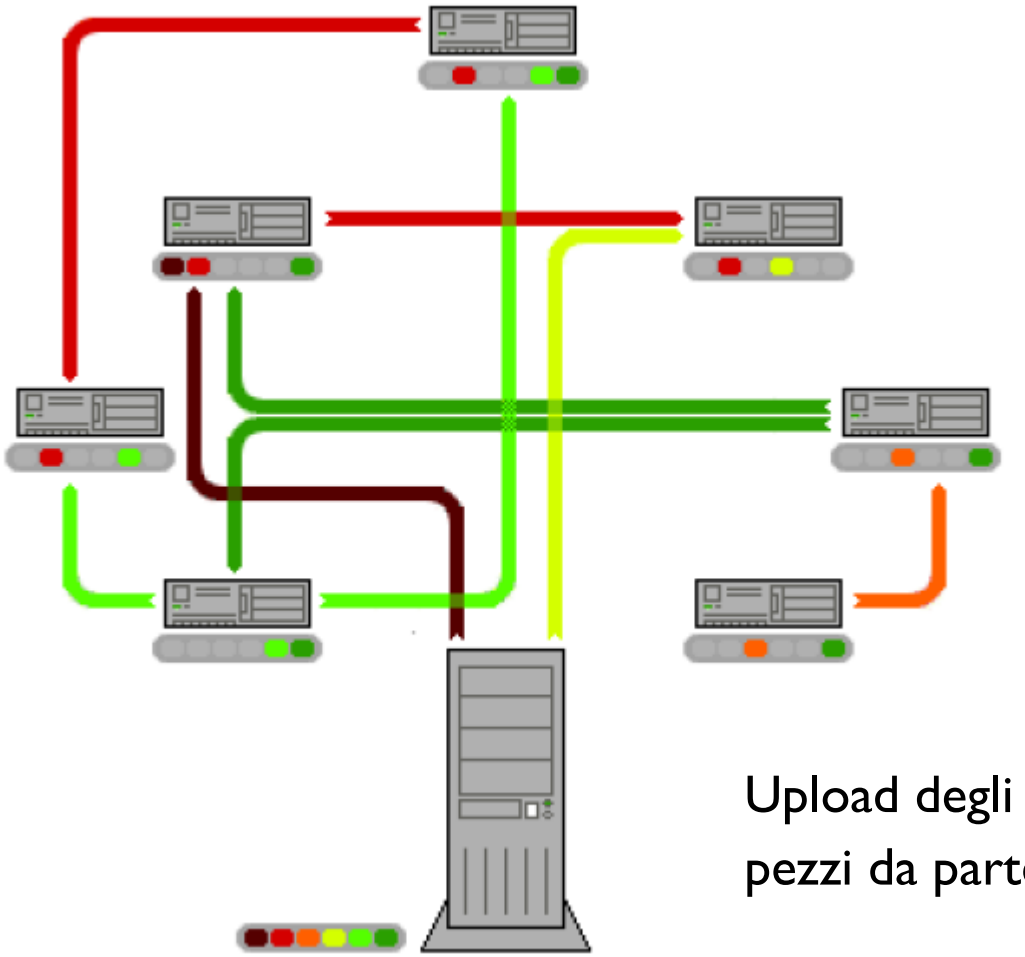
Il server distribuisce i primi 2 due pezzi del file

CONTENT DISTRIBUTION NETWORKS



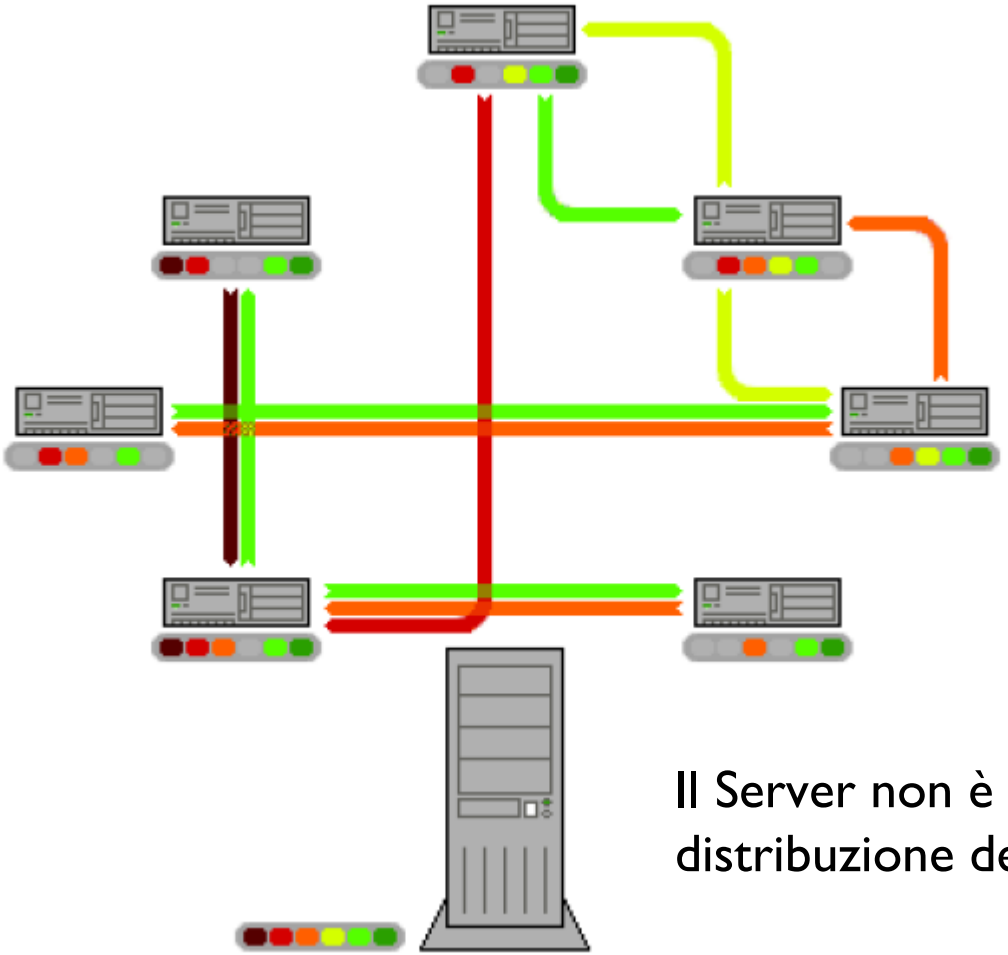
Il server distribuisce altri due pezzi: in parallelo, anche i due pezzi precedenti vengono distribuiti

CONTENT DISTRIBUTION NETWORKS



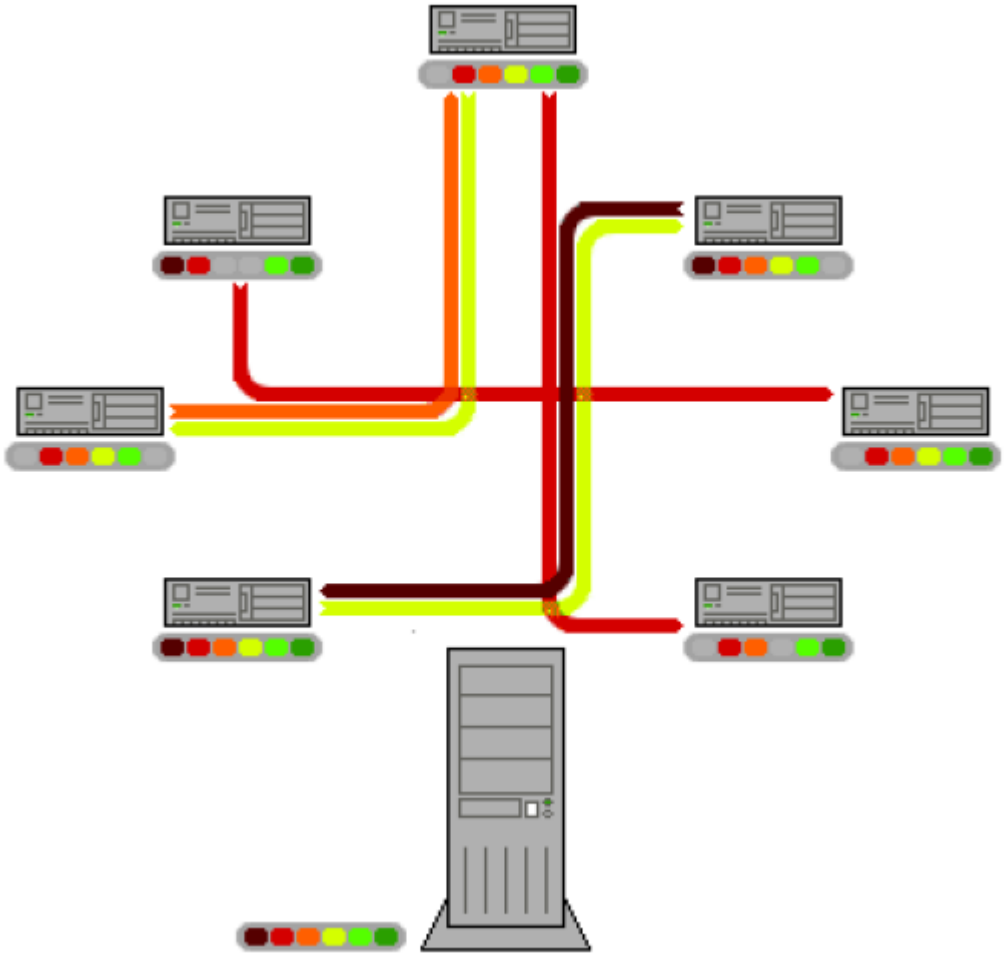
Upload degli ultimi due pezzi da parte del servers

CONTENT DISTRIBUTION NETWORKS

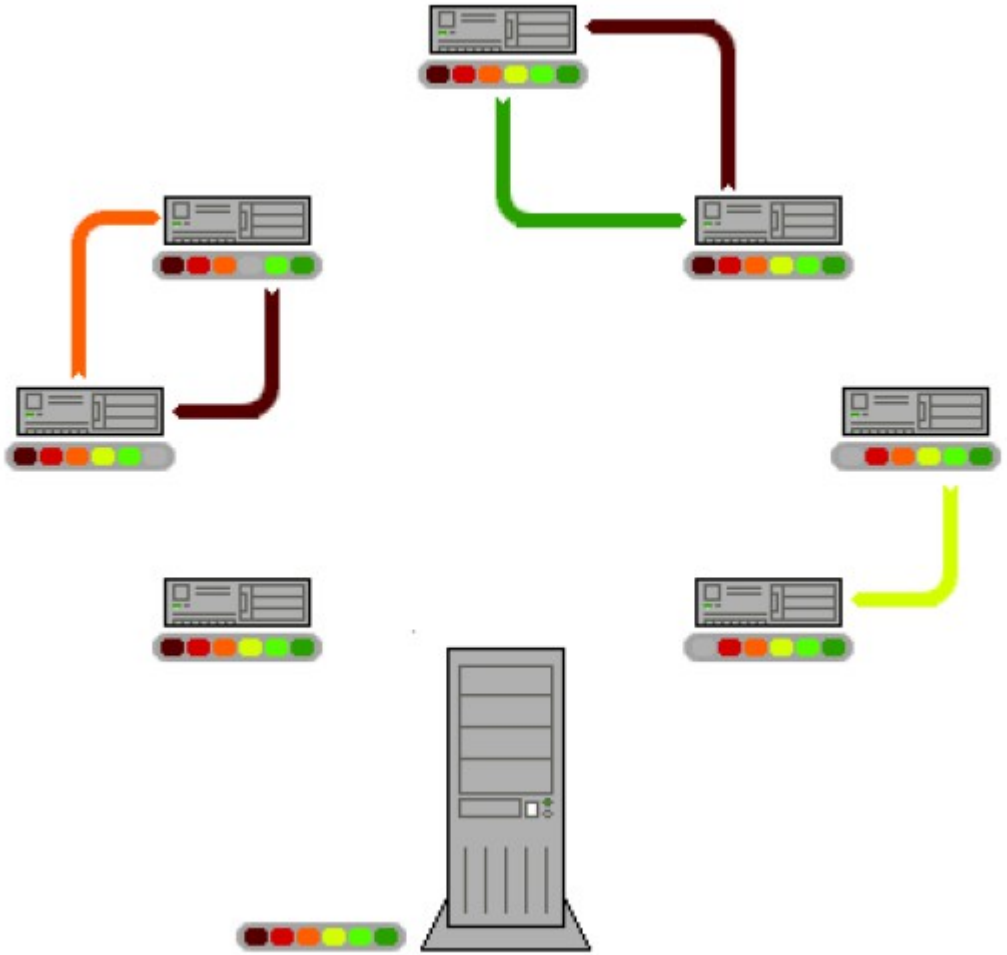


Il Server non è più coinvolto nella distribuzione del file

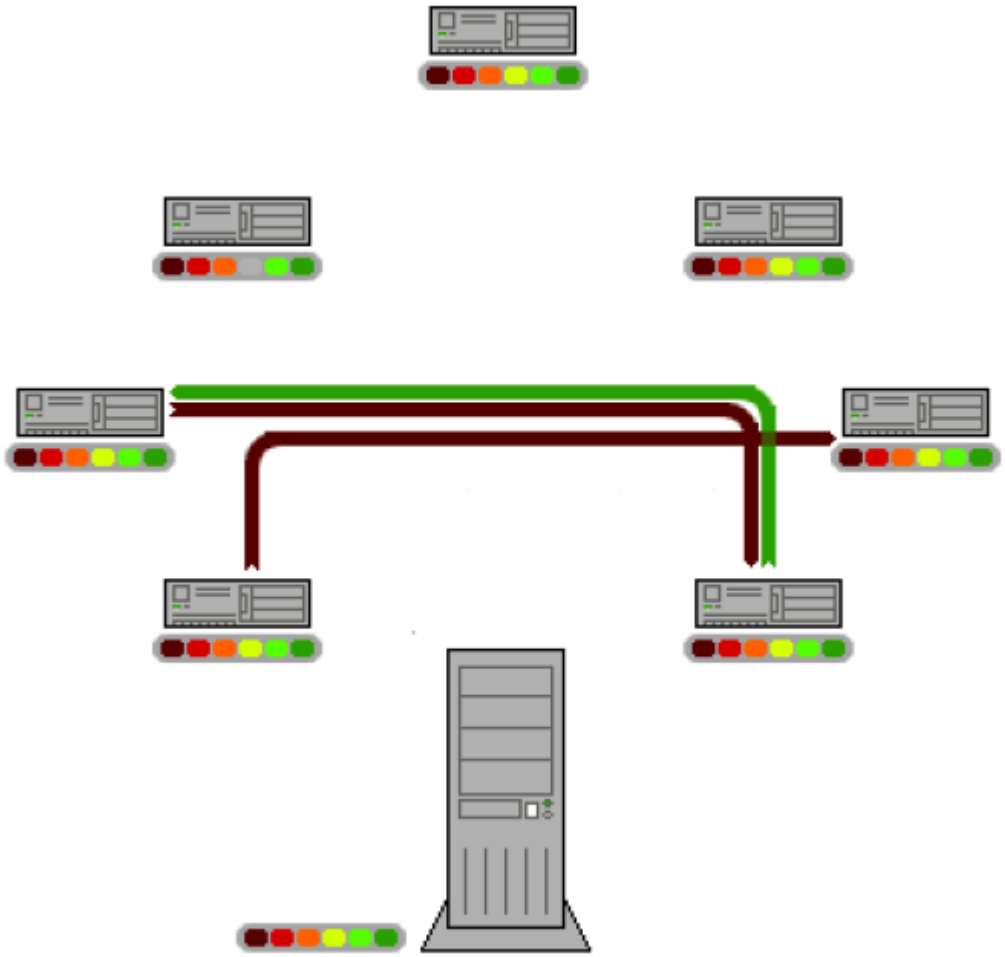
CONTENT DISTRIBUTION NETWORKS



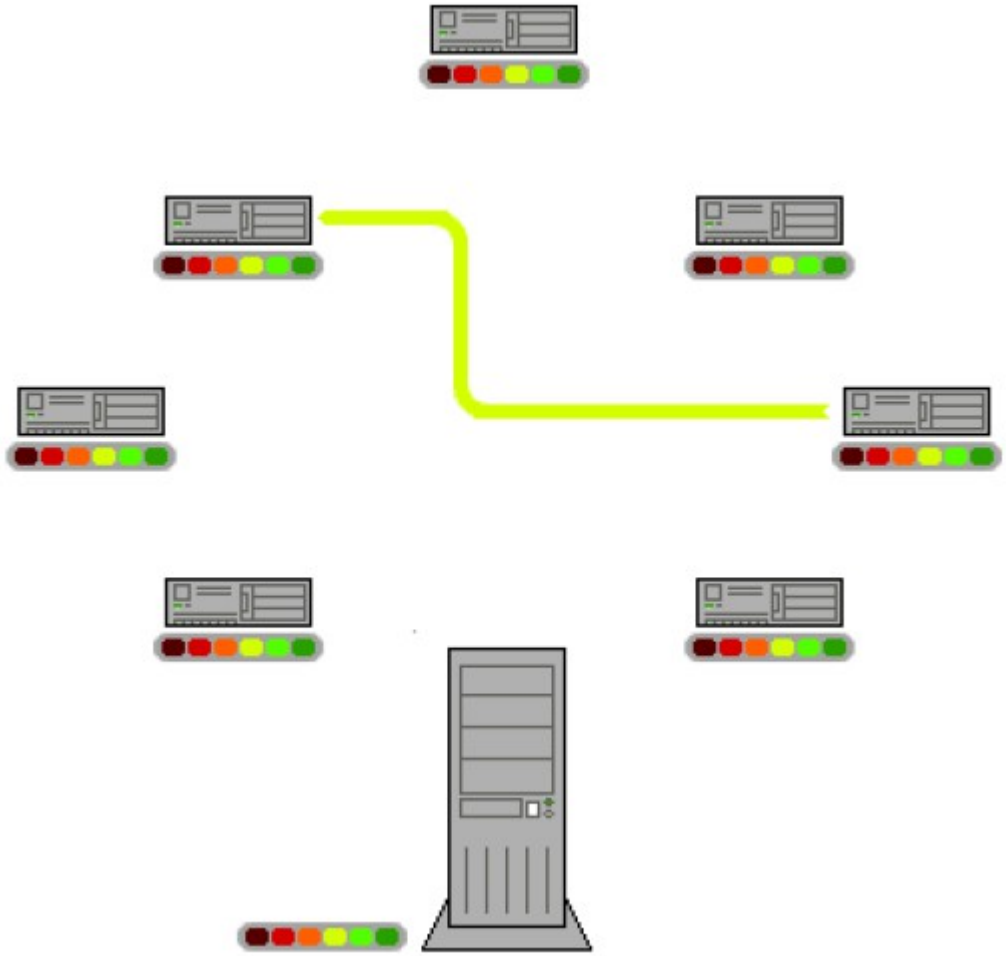
CONTENT DISTRIBUTION NETWORKS



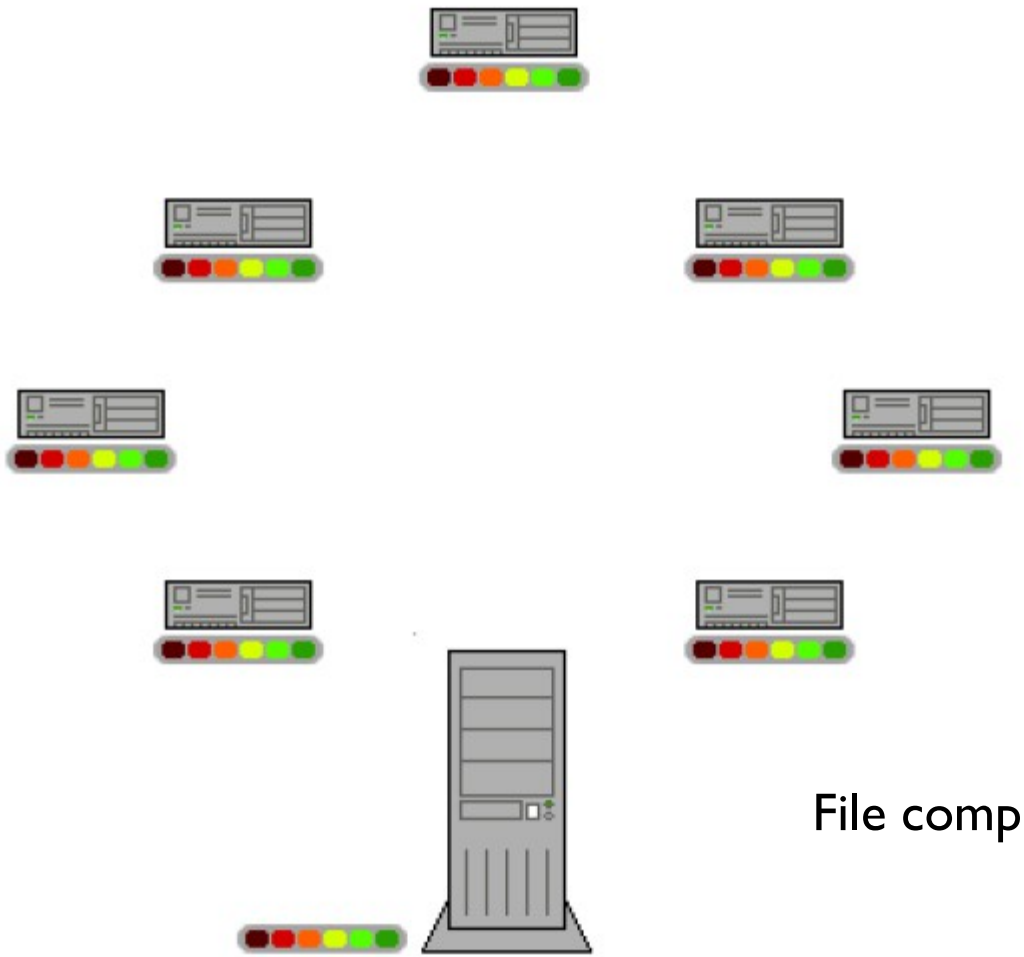
CONTENT DISTRIBUTION NETWORKS



CONTENT DISTRIBUTION NETWORKS



CONTENT DISTRIBUTION NETWORKS



File completo su tutti i peer

BITTORRENT

- BitTorrent è un Sistema per la distribuzione e condivisione di file.
- scritto in Python da Bram Cohen nel 2002: il software è completamente free e Open Source.
- cerca di combattere il fenomeno del free-riding, implementando all'interno dei suoi algoritmi una strategia tit-for-tat.
- è stato uno dei sistemi P2P più diffusi ed è responsabile di molta parte del traffico Internet
- nasce dall'ultima esperienza lavorativa di Bram in un'azienda, dove si conservavano i file dei progetti divisi in parti crittografate, distribuite fra vari computer.
- adotta tecniche tipiche delle content distribution networks

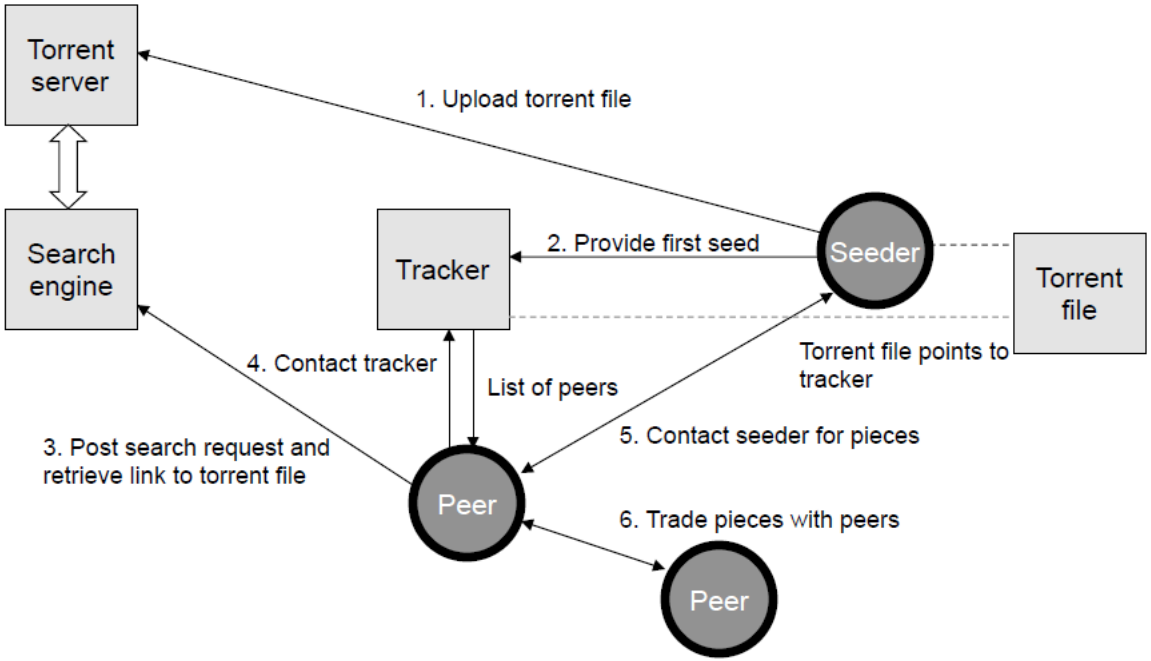
BITTORRENT: GLOSSARIO

- il peer che vuole condividere il file genera il descriptor del file, il .torrent, e lo pubblica su un server.
- il descrittore include il riferimento ad un tracker, un'entità attiva che coordina la condivisione del file sui peer.
- swarm: insieme dei peer che collaborano alla distribuzione dello stesso file, coordinati dallo stesso tracker.
- seeder: peer che possiede tutte le parti di un file
  - all'inizio esiste un singolo seeder S, il peer che pubblica il contenuto, crea il file .torrent e lo pubblica
  - se S lascia la rete dopo aver "inseminato" un insieme di peer, altri peers che posseggono tutto il file divengono seeders. Sono i peer che hanno effettuato il download di tutto il file e sono sempre online
- leecher:
  - il peer che possiede una parte oppure nessuna parte del file e scarica il file dal seeder oppure da altri leachers
  - all'inizio, cerca il file .torrent per iniziare il download

BITTORRENT: FUNZIONAMENTO

- Idea chiave
  - "la popolarità corrisponde alla località temporale" (Flash Crowds)
    - esempio CNN l'11 settembre, release di nuovi film/giochi
  - ottenere una distribuzione efficiente del contenuto mediante file swarming
- la versione iniziale di Bittorrent non svolge tutte le funzioni di un sistema P2P, non supporta ad esempio la ricerca dei file
  - più tardi introdotta la DHT Kademlia per la ricerca dei file
- focus iniziale su download efficiente, non sull'indicizzazione del contenuto
  - incrementa il numero di download, uso efficiente della banda
  - single publisher, multiple downloaders
- diversi implementazioni "legali"
  - Blizzard Entertainment usa questo sistema per distribuire la versione beta dei loro giochi

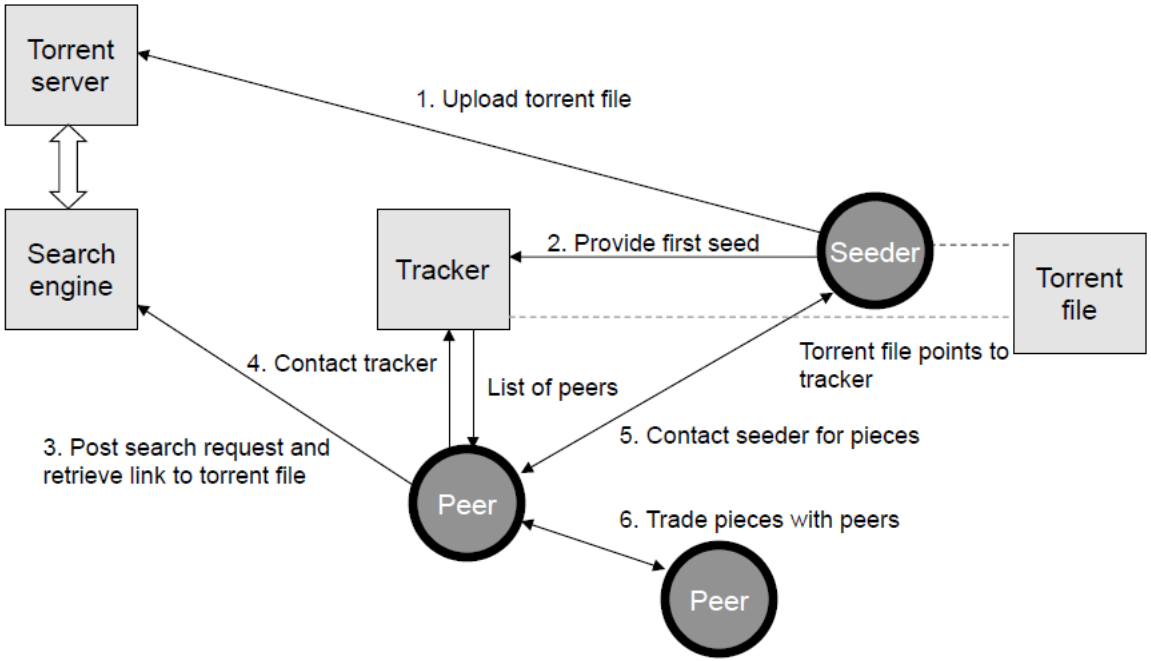
PROTOCOLLO: ARCHITETTURA (I)



- Il Seeder vuole condividere un file

  1. effettua l'upload del .torrent su un Torrent Server
  2. apre una connessione con il tracker e lo informa della propria esistenza: per il momento è il solo peer che possiede il file

PROTOCOLLO: ARCHITETTURA (I)



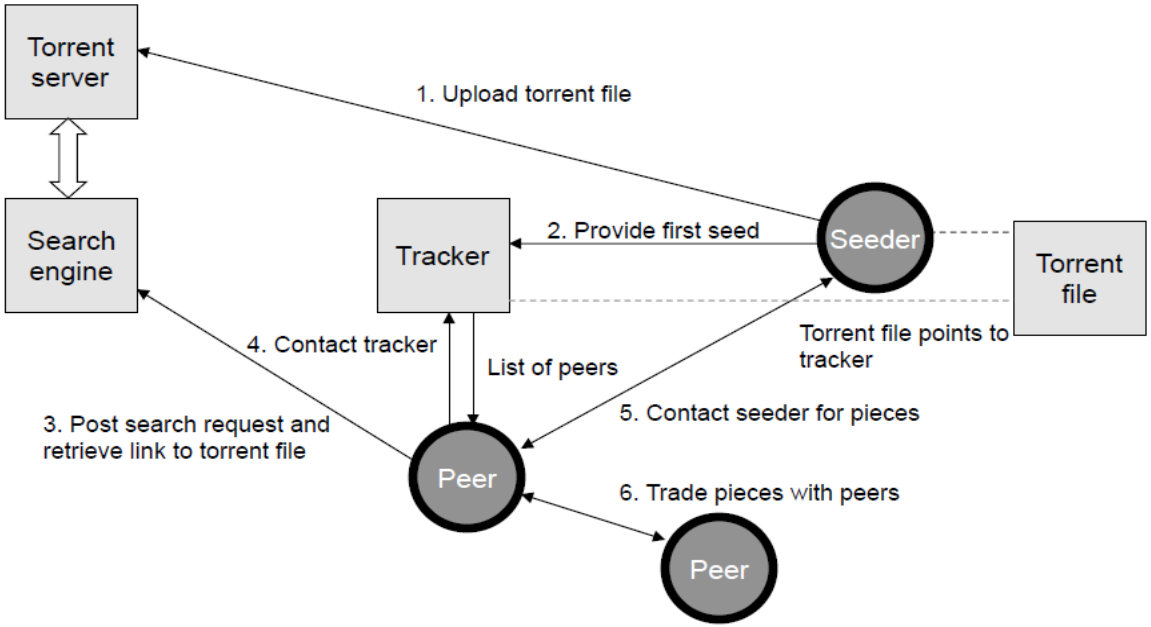
- il leacher (Peer) vuole scaricare un file

  3. scarica il descrittore del file (.torrent) e lo apre tramite un client BitTorrent

  4. apre una connessione verso il tracker, lo informa della propria esistenza e riceve dal tracker la lista dei peer nello swarm

  5. + 6. apre un insieme di connessioni con i peer dello swarm

PROTOCOLLO: ARCHITETTURA (I)



- il leacher (Peer) vuole scaricare un file

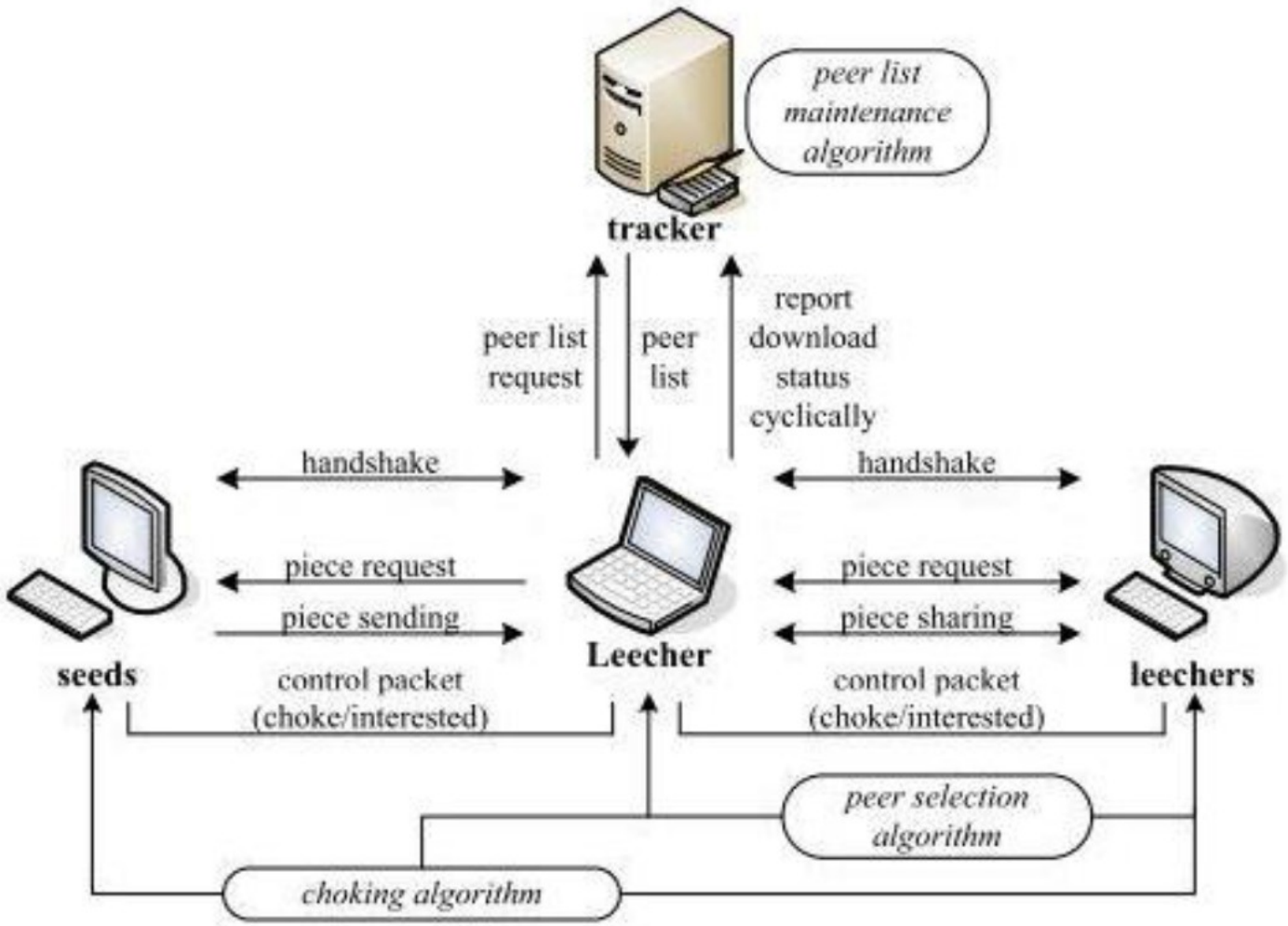
  3. scarica il descrittore del file (.torrent) e lo apre tramite un client BitTorrent

  4. apre una connessione verso il tracker, lo informa della propria esistenza e riceve dal tracker la lista dei peer nello swarm

  5. + 6. apre un insieme di connessioni con i peer dello swarm

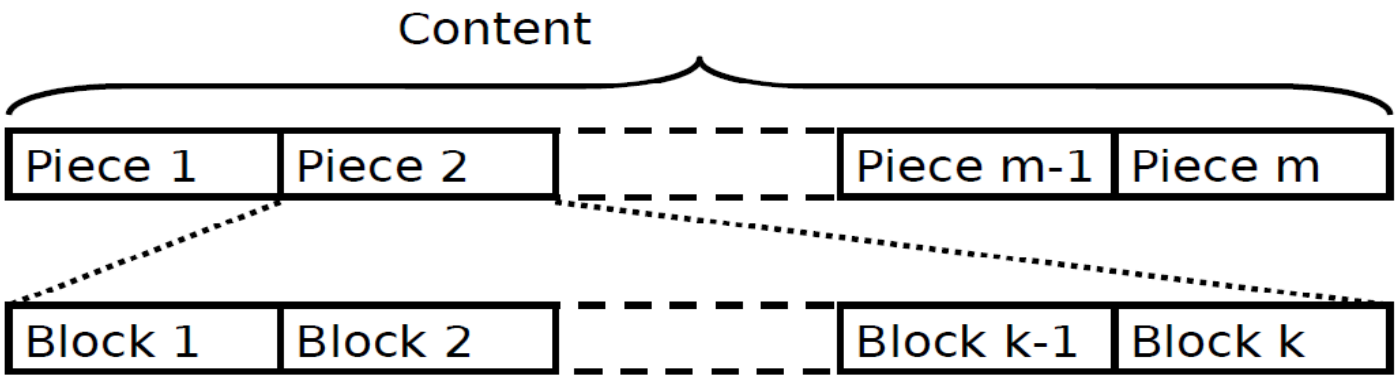
- se il peer rimane online, dopo aver finito il download, continua a distribuire il file, diventando così un seeder

PROTOCOLLO: ARCHITETTURA (2)



PEZZI E BLOCCHI DI FILE

- Il contenuto è diviso in pezzi (256KB - 2MB)
  - la più piccola unità di scambio tra peer: l'idea è di trasferire pezzi tra peer diversi contemporaneamente.
  - ogni volta che un pezzo è completamente scaricato, viene confrontato il suo hash con l'hash contenuto nel .torrent
    - tipicamente 256/512/1024/2048 kByte
    - un hash SHA-1 per pezzo nel file .torrent
    - la dimensione del pezzo adattata per avere un file .torrent di dimensione ragionevole
- I pezzi sono divisi in blocchi (16KB) (unità di trasferimento)



IL FILE .TORRENT

- un qualsiasi client BitTorrent può sia leggere file.torrent sia generarli
- Info key
  - lunghezza del contenuto in bytes
  - md5 hash del contenuto (opzionale)
    - non usata dal protocollo
    - sufficienti gli hash SHA-1 per il controllo dei pezzi
  - Nome file
  - Lunghezza dei pezzi (256kB, 512kB, 1024kB, etc.)
  - Concatenazione di tutti gli hash SHA-1 dei pezzi
  - URL del tracker (HTTP)
    - possibilità di annunciare una lista di backup trackers
  - Alcuni campi opzionali:
    - data di creazione, comment, creato da,....

IL FILE .TORRENT

- generato automaticamente da tool opportuni (maketorrent) o direttamente dal client Bittorrent
- al momento della generazione, il tool richiede al client:
  - tutte le informazioni per create il .torrent
  - in particolare l'indirizzo di un tracker.
    - scegliere l'indirizzo del tracker da una lista predefinita
    - inserire l'indirizzo di un tracker conosciuto ad esempio: http://www.smarttorrent.com:2710/announce

```
<?xml version="1.0" encoding="UTF-8"?>
<tor:TORRENT
   xmlns:tor=http://azureus.sourceforge.net/files
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://azureus.sourceforge.net/files
           http://azureus.sourceforge.net/files/torrent.xsd">
 <ANNOUNCE_URL>http://torrent.opera.com:6969/announce</ANNOUNCE_URL>
 <CREATION_DATE>1281694453</CREATION_DATE>
 <TORRENT_HASH>2EEAB52F0242C42AC788D17924CCC197DC3D2F0C</TORRENT_HASH>
 <INFO>
  <NAME encoding="utf8">Opera_1061_en_Setup.exe</NAME>
  <PIECE_LENGTH>262144</PIECE_LENGTH>
  <LENGTH>10809256</LENGTH>
  <PIECES>
   <BYTES>54E6CCEC9EB3BEA12711FB9F383890023650D5AF</BYTES>
   <BYTES>D134F302346EFB60E57BA240F8F38BD27A408C35</BYTES>
   <BYTES>73811C5D1313F5A16A18BFF77DCAD18CADFB3F97</BYTES>
    .
    .
    .
   <BYTES>6AE8353393A66804FF95C98624331824A6F61D62</BYTES>
  </PIECES>
 </INFO>
</tor:TORRENT>
```

PEER BOOTSTRAP

Il client peer scarica il .torrent file da un web server

- reperisce la URL del tracker e si connette ad esso (HTTP GET)
- Invia al tracker
  - Info_hash (SHA-1 of the info key)
  - Peer ID
  - Porta su cui è in ascolto il client
  - Numero di peer richiesti (default 50)
  - Periodicamente invia al tracker statistiche varie
    - Uploaded, downloaded, left, event (started, stopped, completed)
- il tracker restituisce
  - una lista di peer dello swarm scelti in modo casuale
    - peer ID, peer IP, porta
    - tipicamente 50 peers
  - statistiche
    - complete/incomplete (seed/leechers)

PEER BOOTSTRAP

- Il Tracker
  - non è coinvolto nella distribuzione del file
  - è un web server che svolge anche funzione di "certification authority" per i file
    - l'hash SHA-1 di ogni pezzo è nel file .torrent
    - il peer sfrutta il .torrent ricevuto dal Tracker per controllare l'integrità dei pezzi ricevuti dalla rete P2P

- Il peer si connette ad un sottoinsieme di peer restituiti dal Tracker
  - al massimo k connessioni in uscita
  - i rimanenti peer sono tenuti in un pool per connettersi se gli altri si disconnettono

- neighbour peer set: lista dei peer conosciuti P
  - ricevuti dal tracker
  - contattati dal peer

- Risultato: un grafo denso di connessioni tra peer (obiettivo: un random graph)

STATO DEI PEER

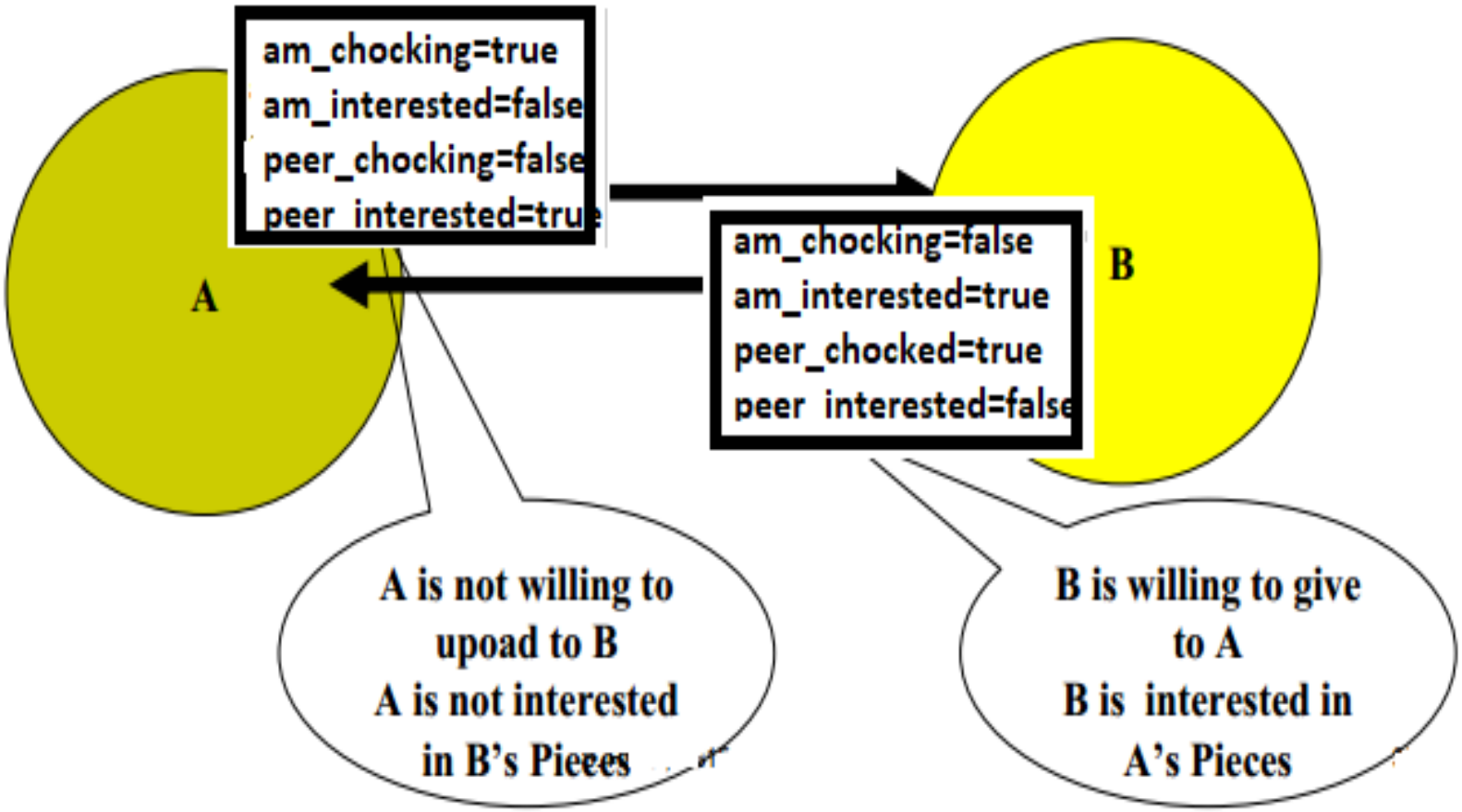
Ogni peer mantiene, per ogni peer remoto a cui è connesso, il suo stato:

- am_choking**:** il peer locale sta "strozzando" la comunicazione verso il peer remoto. Non vuole inviare dati al peer remoto
- am_interested**:** il peer locale è interessato in almeno un pezzo posseduto dal peer remoto.
- peer_choking**:** il peer remoto sta "strozzando" la comunicazione verso il peer locale. Non vuole inviare dati al peer locale
- peer_interested: il peer remoto è interessato in almeno un pezzo posseduto dal peer locale

Il peer locale può ricevere dati dal peer remoto se:

- il peer locale è interessato nel peer remoto
- lo stato del peer locale nel peer remoto è unchocked

STATO DEI PEER

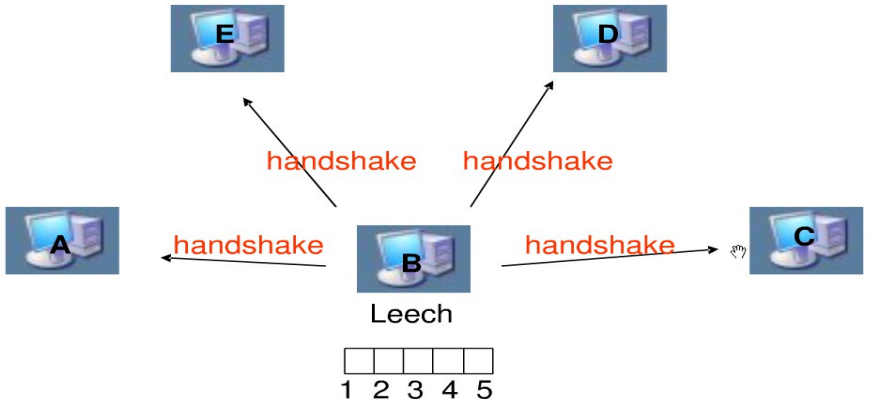


PEER WIRE PROTOCOL

- il protocollo peer to peer definisce le regole per lo scambio dei pezzi dei file
- a differenza del modello client server, un peer che riceve una richiesta di connessione può sempre rifiutarla
  - la decisione di effettuare l'unchoke di un peer è presa dall'algoritmo di choke/unchoke
  - inoltre un peer può rifiutare la richiesta di connessione se:
    - l'hash ricevuto non corrisonde all'hash di alcuno dei file posseduti
    - l'ID del peer che chiede la connessione non corrisponde ad alcuno degli ID ricevuti dal tracker
- non appena la connessione viene stabilita, ogni peer comunica al partner gli indici dei pezzi del file che possiede
- ogni peer dello swarm conosce la distribuzione dei pezzi all'interno dello swarm

PEER WIRE PROTOCOL: MESSAGGI

- **Handshake**
  - two way handshake
  - per iniziare una connessione tra i peers
  - una volta iniziata la connessione è simmetrica
  - contiene (68 bytes)
    - Protocol string identifier length
    - Pstr="BitTorrent protocol"
    - Reserved (8 bytes)
    - Info_hash
    - Peer ID



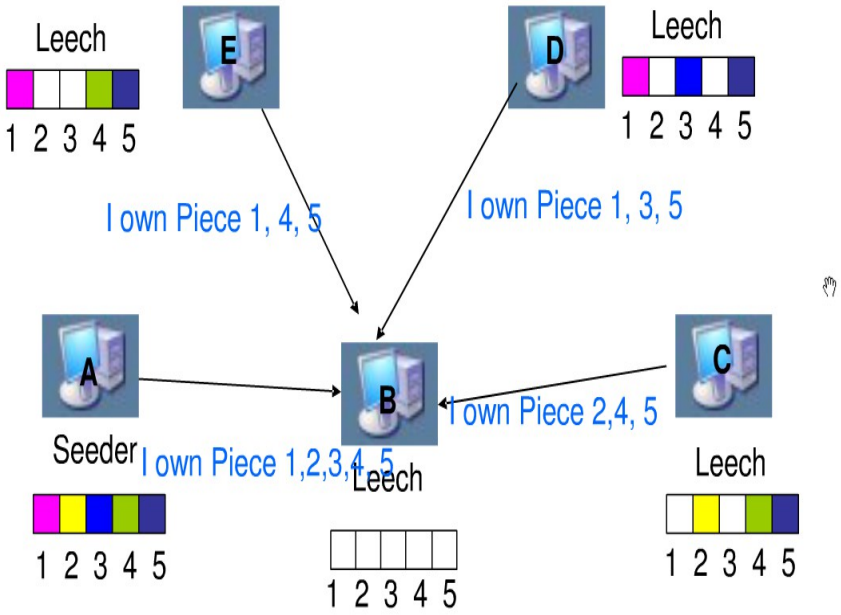
- **Keep Alive**
  - inviato ogni due minuti se non viene ricevuto alcun messaggio su una connessione
  - controlla se la connessione è stata terminata dall'altro capo

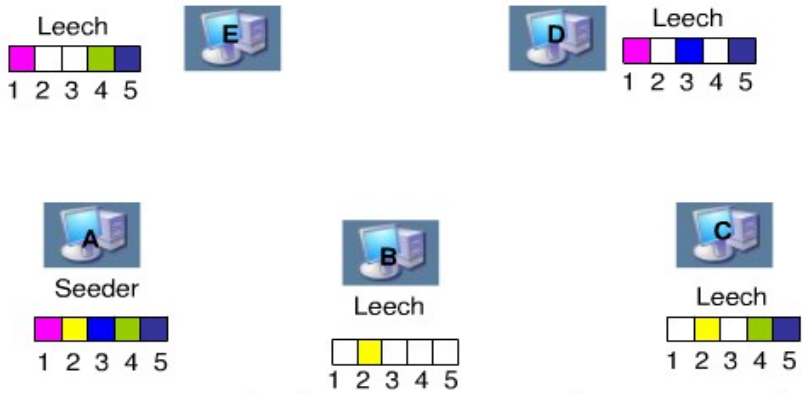
PEER WIRE PROTOCOL: MESSAGGI

- Choke/Unchoke richiesti per implementare l'algoritmo di unchocking
  - choke:
    - inviato da A a B, quando A "effettua il chocking" di B
    - nessuna richiesta da B è accettata da A dal momento del chocking
  - unchoke:
    - inviato da A a B quando A "effettua l'unchoke" di B
- Interested:
  - inviato da A a B quando A è interessato a B
- Not Interested
  - inviato da A a B per indicare che A non è interessato a B

PEER WIRE PROTOCOL: MESSAGGI

- Bitfield <bitfield>
  - il primo messaggio inviato dopo l'handshake
    - non inviato più in seguito
    - inviato da entrambi i peer una volta che la connessione è inizializzata
  - Bit i nel bitfield è settato a 1 se il peer possiede il pezzo I, 0 altrimenti





Soon,
"B" summarized the pieces available in its swarm.

piece	Total number in the swarm of B	Corresponding peers
1	3	(A, D, E)
2	2	(A, ,B, C)
3	2	(A, D)
4	3	(A, C, E)
5	4	(A, C, D, E)

PEER WIRE PROTOCOL: MESSAGGI

Messaggi per lo scambio dei blocchi:

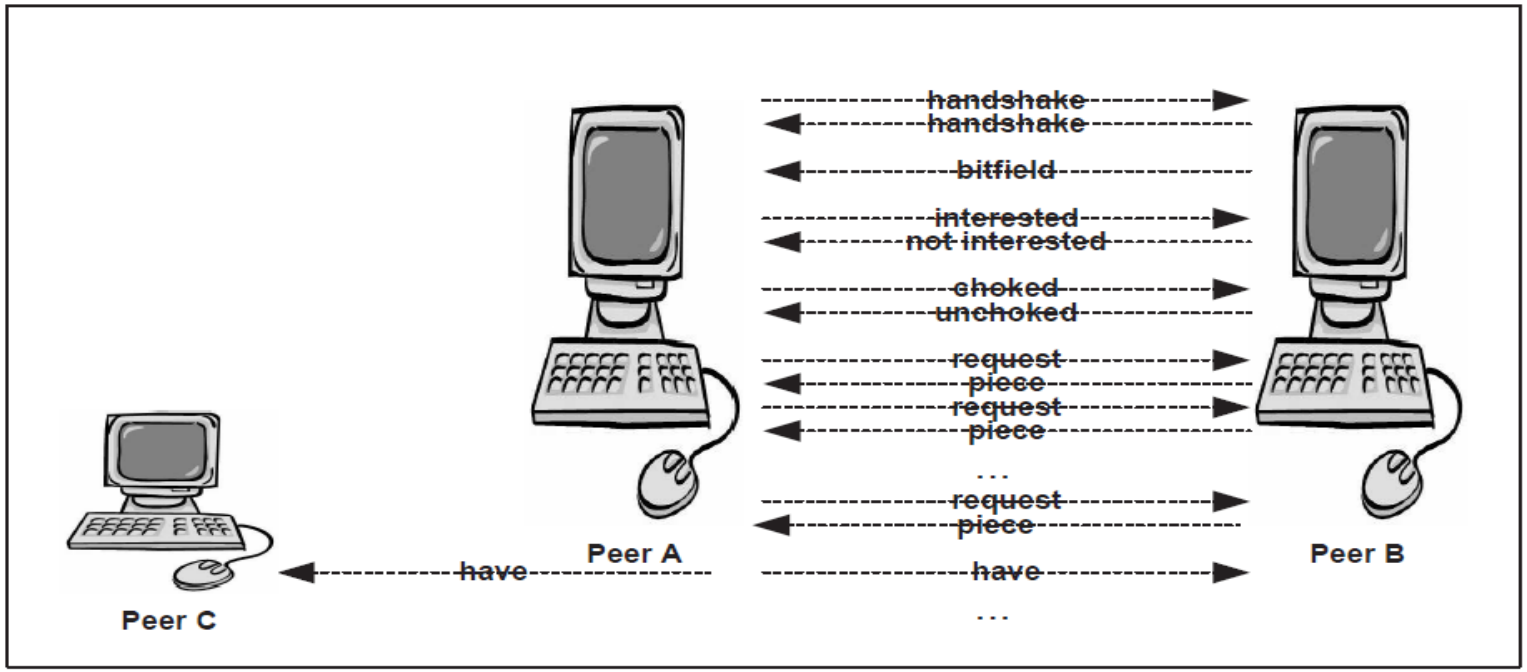
- Request <index><begin> <length>
  - inviato dal peer A al peer B per richiedere al peer B il blocco
    - del pezzo di indice <index>
    - che inizia all'offset <begin> dentro il pezzo
    - di lunghezza <length>
- Piece <index> <begin> <block>
  - messaggio utilizzato per inviare i blocchi
  - mandato dal peer A al peer B per inviare un blocco di dati
    - del pezzo di indice <index>
    - che inizia all'offest <begin> all'interno del pezzo
    - payload è <block>

PEER WIRE PROTOCOL: MESSAGGI

- Have
  - notifica l'acquisizione di un nuovo pezzo.
    - inviato a tutti i peer nel Peer Set
  - informazione utile per monitorare lo swarm: conoscere esattamente chi ha cosa (quali pezzi).

- Cancel
  - usato solo nella fase di end game mode
  - inviato dal peer A al peer B per cancellare una richiesta di blocco precedentemente inviata a B
    - del pezzo di indice <index>
    - che inizia all'offeset <begin> all'interno del pezzo
    - di lunghezza <length>

PEER WIRE PROTOCOL



COME VENGONO SELEZIONATI I PEER ED I PEZZI

- Algoritmi per la selezione dei peer
  - Chocking/ Unchocking (basati sulla teoria dei giochi)
    - ricompensa dinamicamente i peer più "collaborativi" peers, quelli che hanno dato un numero magiore di contenuto nel passato
  - Optimistic Unchocking

- Algoritmi per la selzione dei pezzi dei files
  - Strict Priority
  - Random First Piece
  - Rarest First
  - End Game

IL PROBLEMA DEI FREE RIDERS

- Free rider: un individuo che
  - nasconde la propria preferenza per un bene comune
  - evitando così di pagare il suo prezzo
  - attribuisce agli altri il compito di pagare il bene comune
- il free rider è consapevole che può beneficiare del bene comune senza pagarlo, purchè ci sono individui interessati nel bene comune,
  - è difficile escludere l'individuo dall'utilizzare il bene comune.
- esempio: un gruppo di studenti devono decidere se comprare una TV per l'appartamento comune
  - qualcuno dice che non gli/le interessa una TV per evitare il pagamento del prezzo
  - quando il bene è stato acquistato, non è facile impedire al free rider di utilizzarlo
- il nome deriva dalla persona che monta sull'autobus senza pagare il biglietto.

FREE RIDERS IN BITTORRENT

- Free riders in BitTorrent:
  - peer che non mettono a disposizione della comunità la loro banda di upload
- un buon comportamento della rete Bittorrent dipende dal "comportamento cooperativo" dei peer rispetto alla comunità
  - necessaria la eliminazione dei "free riders"
- un approccio possibile per risolvere questo problema è basato sulla reciprocità:
  - un client ottiene un buon servizio se e solo se fornisce un buon servizio alla comunità
  - "forzare" il peer "egoista" ad avere un buon comportamento
  - non può essere risolto semplicemente "codificando strategie oneste" nel codice del client

FREE RIDERS IN BITTORRENT

- la soluzione è complessa, perchè
  - non esiste una entità centralizzata che controlli i nodi
  - non è possibile imporre un certo comportamento ai client
    - reverse engineering per modificare il client
  - molti client non officiali di Bittorrent permettono all'utente di limitare la banda in upload a piacimento.

- l'approccio di Bittorrent
  - una strategia basata sul monitoraggio dinamico delle connessione
    - calcola la priorità dei peer dinamicamente, dando una priorità maggiore ai vicini che inviano dati ad una frequenza maggiore
  - idee derivate dalla teoria dei giochi
    - strategia Tit for Tat , basata sul problema del prigioniero iterato
    - Choke algorithm

NEIGHBOUR (ED ACTIVE) SET

- ogni peer mantiene un neighbour peer set composto da un insieme di altri peer nello swarm
  - ritrovato inizialmente dal tracker
  - massima dimensione di questo insieme in genere 80
  - se |neighbors| < soglia, vengono chiesti al Tracker nuovi peer
- il peer apre un insieme di connessioni TCP con un sottoinsieme di peer nel suo neighbour set
  - lascia un insieme di slot aperti per accettare connessioni dagli altri peer
- i peer unchocked (non soffocati) formano l'active neighbour set
  - aggiornato periodicamente con l'esecuzione dell'algoritmo di choke
  - peer selection: l'algoritmo seleziona i vicini a cui il peer locale invierà i dati (upload)

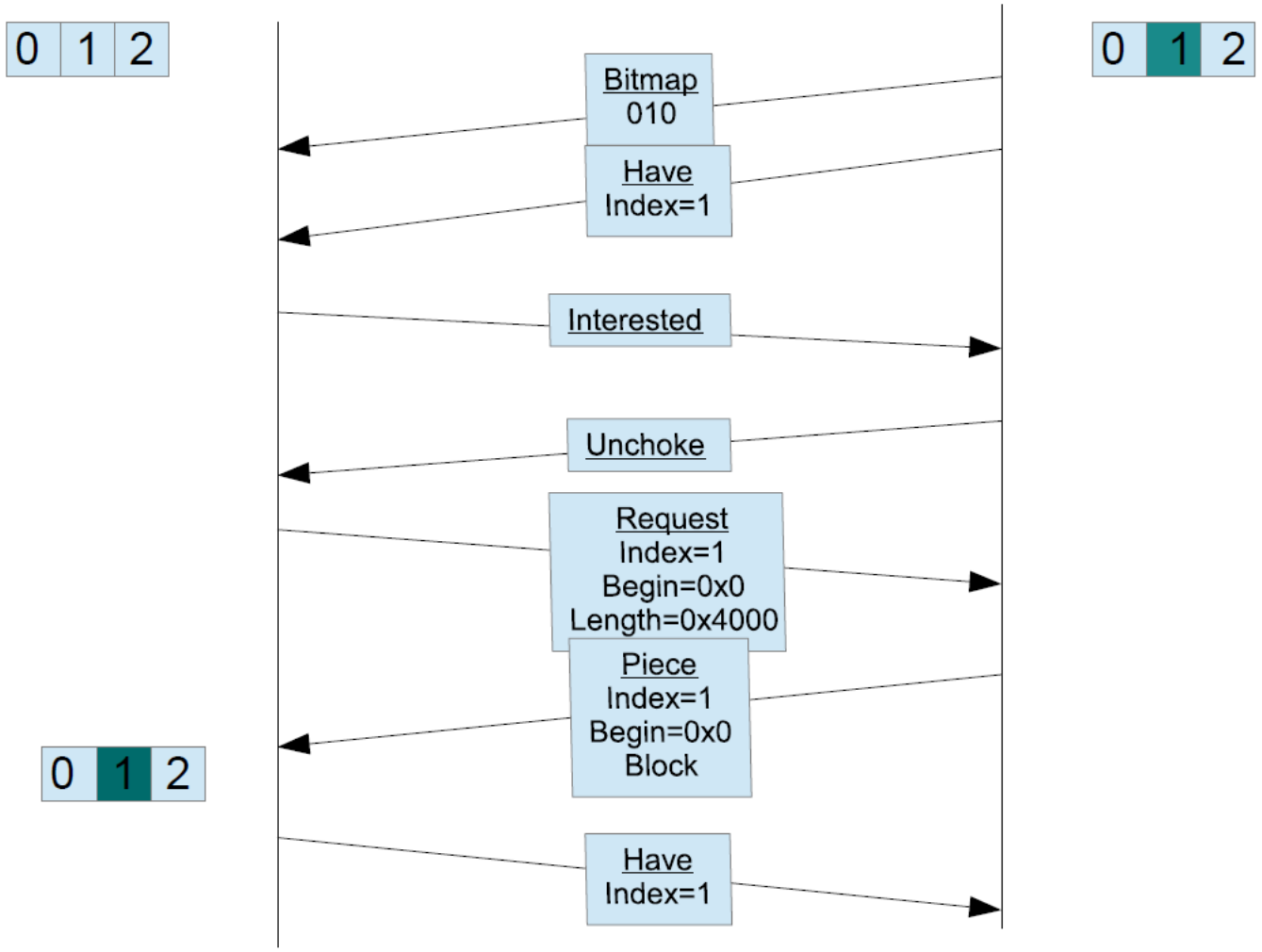
CHOCKING IN BREVE

- BitTorrent divide il tempo in round
  - per ogni round, decide a chi inviare dati (upload)/ da chi scaricare i dati (download)
  - round di 10 secondi
- Ogni connessione con un peer nel neighbour set è controllata da 4 stati
  - Interested / uninterested: voglio o non voglio un pezzo da te?
  - Chocked/unchocked: sto scaricando dati da te?
- Le connessioni (TCP) sono bidirezionali, quindi:
  - il peer locale decide se è interessato/se vuole soffocare il peer remoto
  - il peer remoto decide se è interessato nel peer locale/se lo vuole soffocare

CHOCKING IN BREVE

- tutte le connessioni iniziano nello stato 'Not Interested' and 'Choked'.
- per raggiungere uno stato in cui il peer locale possa ricevere files:
  - il peer locale deve inviare un messaggio "Interested" ad un peer da cui si vuole effettuare il download
    - "vorrei richiedere e scaricare dei pezzi da te"
  - il peer locale deve attendere un messaggio di "Unchoke" dal peer remoto
    - dopo aver inviato un messaggio "Interested", aspetta l'unchoke prima di inviare qualsiasi richiesta
    - una volta ricevuto l'unchoke può inziare a richiedere i pezzi (messaggio "Request")
      - i pezzi di solito sono troppo grandi per essere inviati singolarmente
      - invio di singoli blocchi del pezzo

CHOCKING IN BREVE



LEECHER CHOKE ALGORITHM

- come si individuano i peer con cui "commerciare" i pezzi?
- meccanismo di incentivi
  - basato sulla strategia tit-for-tat
  - "se mi dai un pezzo, io ti darò un pezzo, altrimenti non ti darò niente"
- due meccanismi di base
  - regular choking ed optimistic unchoke
- meccanismo di choking: rifiuto di upload
  - confinare i free riders
  - limitare il numero di connessioni aperte simultaneamente
- algoritmi di chocking diversi per
  - leecher
  - seeder

LEECHER CHOKE ALGORITHM

- quando è chiamato l'algoritmo?
  - periodicamente, ogni 10 secondi (round)
- inoltre viene invocato tutte le volte che:
  - un peer che è unchoked ed interested lascia il peer set
  - un peer che è unchoked cambia stato da interested a non interested o viceversa
  - migliora la reattività
- ogni 3 rounds (30 secondi)
  - viene selezionato un peer interested e chocked, in modo casuale
  - planned optimistic unchoke: il suo stato verrà cambiato in unchoked successivamente

LEECHER CHOKE ALGORITHM

- ogni volta che l'algoritmo è invocato si ordinano i peer interested e non snubbed, secondo la frequenza con cui hanno inviato i dati al peer locale
  - snubbed
    - non hanno inviato alcun blocco negli ultimi 30 secondi
    - favorisce i peer che hanno dato qualche contributo nell'ultimo round
- si effettua l'unchoke dei tre 3 peers della lista
- se il planned optimistic unchoke non è uno di questi 3 peer, STOP
- altrimenti... (vedi slide successiva)

LEECHER CHOKE ALGORITHM

- se il planned optimistic unchoke fa parte dei primi 3 selezionati dalla lista
  - si sceglie un altro peer in modo casuale
    - un nuovo planned optimistic unchoke (POU)

  se questo nuovo POU è
    - interested, si effettua l'unchoke di quel peer, STOP
    - not interested, si effettua il suo unchoke e si sceglie un altro POU in maniera casuale e si itera.

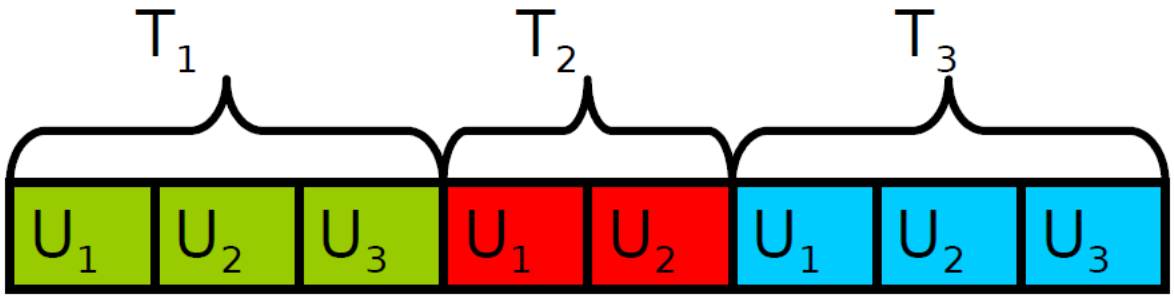
- al più 4 peer possono essere interested ed unchoked !
  - ma...più di 4 possono essere unchoked, dopo la esecuzione di un round
  - nel caso in cui un peer unchoked diventi interested, l'algoritmo viene invocato immediatamente
  - aumenta la reattività nel caso vi siano pochi peer in stato interested

SEEDER CHOKE ALGORITHM

- prima versione
  - stesso algoritmo dei leecher, ma i peer sono ordinati in base alla velocità di download dal seeder
  - i peer con una maggiore banda monopolizzano il seeder!
    - anche se non contribuiscono alla diffusione del contenuto.....
    - non aiuta ad eliminare i free riders
- per questo motivo, in seguito introdotta una seconda versione:
  - algoritmo Round Robin
  - invocato con le stesse regole dell'algoritmo leacher
    - ogni 10 secondi + reattività

SEEDER CHOKE ALGORITHM

- peers interested su cui è stato effettuato l'unchoke meno di 20 secondi prima o che hanno richieste pendenti sono ordinati secondo il tempo in cui è stato effettuato l'unchoke
- quelli con unchoke più recente, per primi
  - peers dovrebbero essere attivi o recenti
- la velocità di upload verso quei peer discrimina verso i peer con lo stesso tempo di unchoke

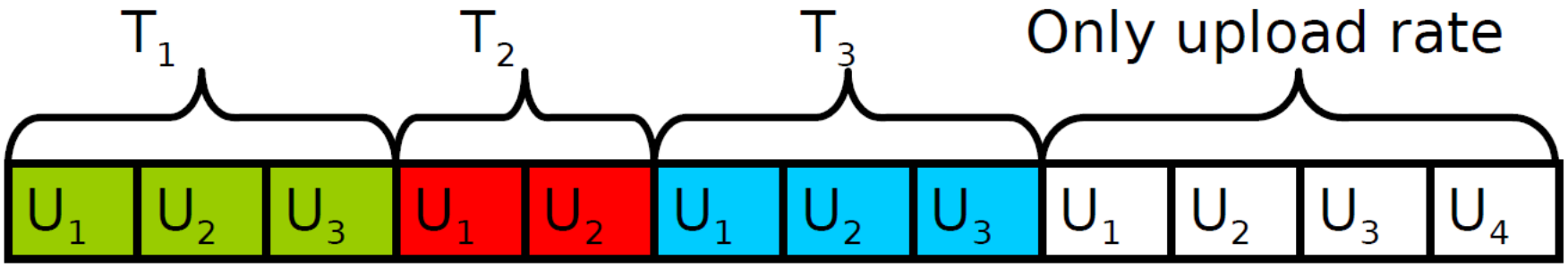


Last unchoked time: $T_1 < T_2 < T_3$

Upload rates: $U_1 > U_2 > U_3$

SEEDER CHOKE ALGORITHM

- tutti gli altri peer unchoked ed interested sono ordinati secondo la velocità di upload, con la priorità più bassa

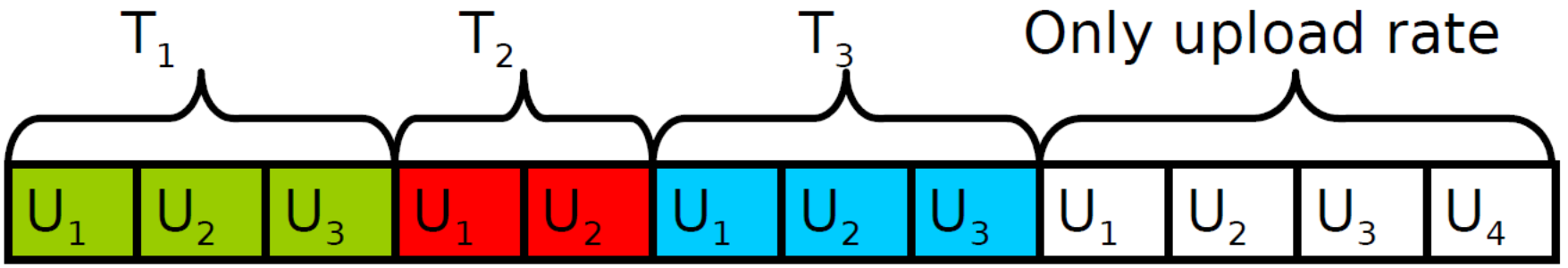


Last unchoked time: $T_1<T_2<T_3$

Upload rates: $U_1>U_2>U_3>U_4$

SEEDER CHOKE ALGORITHM

- per due round su tre, i primi 3 peer sono lasciati unchoked, and un peer che è chocked ed interested viene scelto in modo casuale e ne viene effettuato l'unchoke
- al il terzo round, i primi 4 peer sono lasciati unchoked



Last unchoked time: $T_1<T_2<T_3$

Upload rates: $U_1>U_2>U_3>U_4$

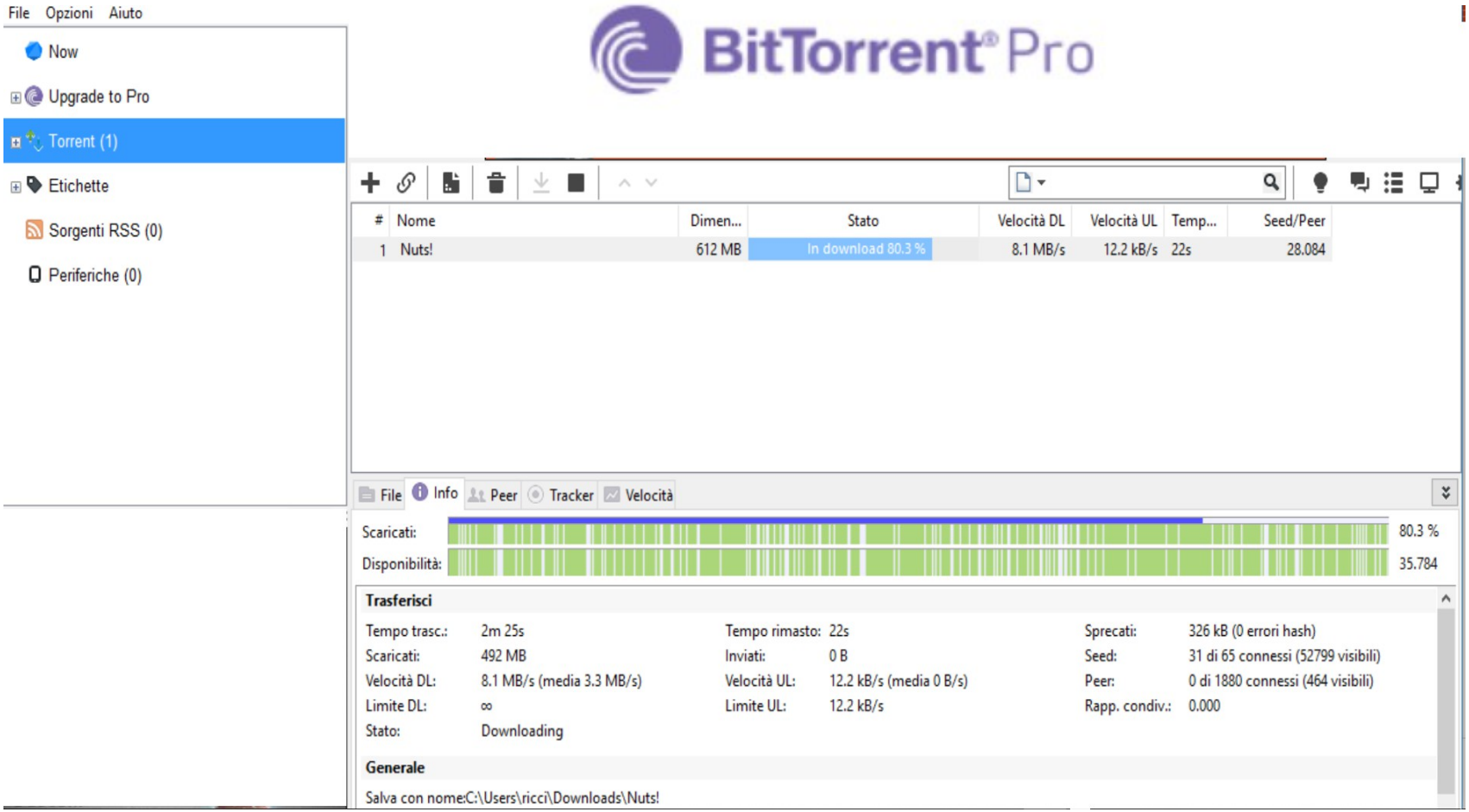
LEECHER CHOKE ALGORITHM: SNUBBING

- se un leecher non riceve alcun blocco da un peer remoto lo "snobba"
  - non gli invia dati, a meno che non sia scelto in un optimistic unchoke
  - può anche chiudere la connessione (dipende dal client)

- un peer è definito snobbato se tutti i peer remoti lo hanno snobbato
  - il peer snobbato non riceve dati da nessuno, di conseguenza, non esegue upload verso nessun altro peer
  - il download dei contenuti è bloccato per il peer snobbato, fino che il peer non diventa target di un optimistic unchock

- Antisnubbing:
  - un peer snobbato aumenta il numero di optimistic unchockes
  - scopo: individuare un peer che alla fine effettui un upload verso esso

BITTORRENT: MONITORARE LE CONNESSIONI

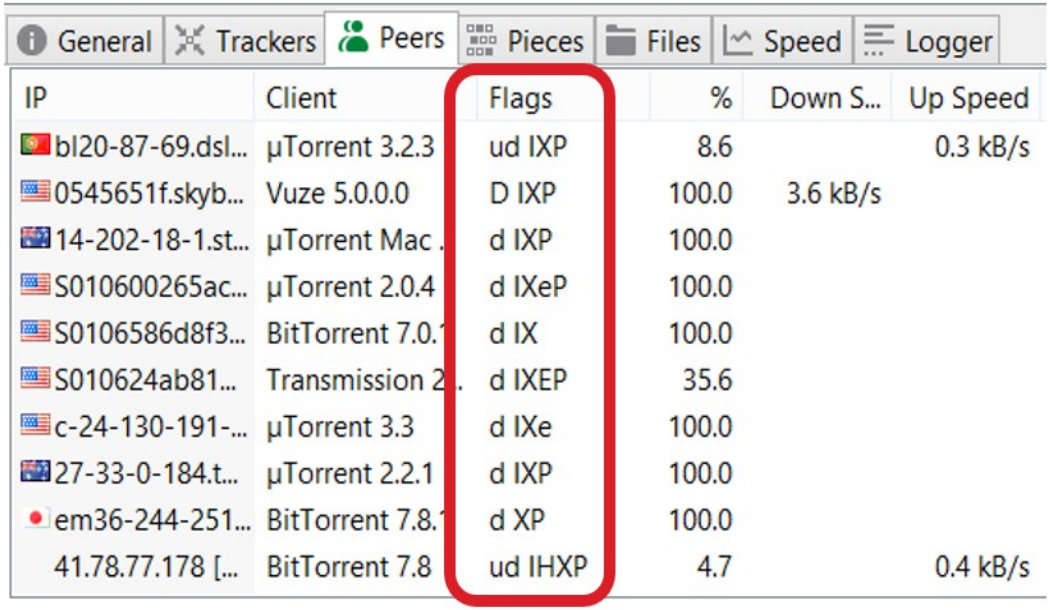


Scaricare client Bittorrent da https://www.bittorrent.com/lang/it/

BITTORRENT: STATO DELLE CONNESSIONI

Download control

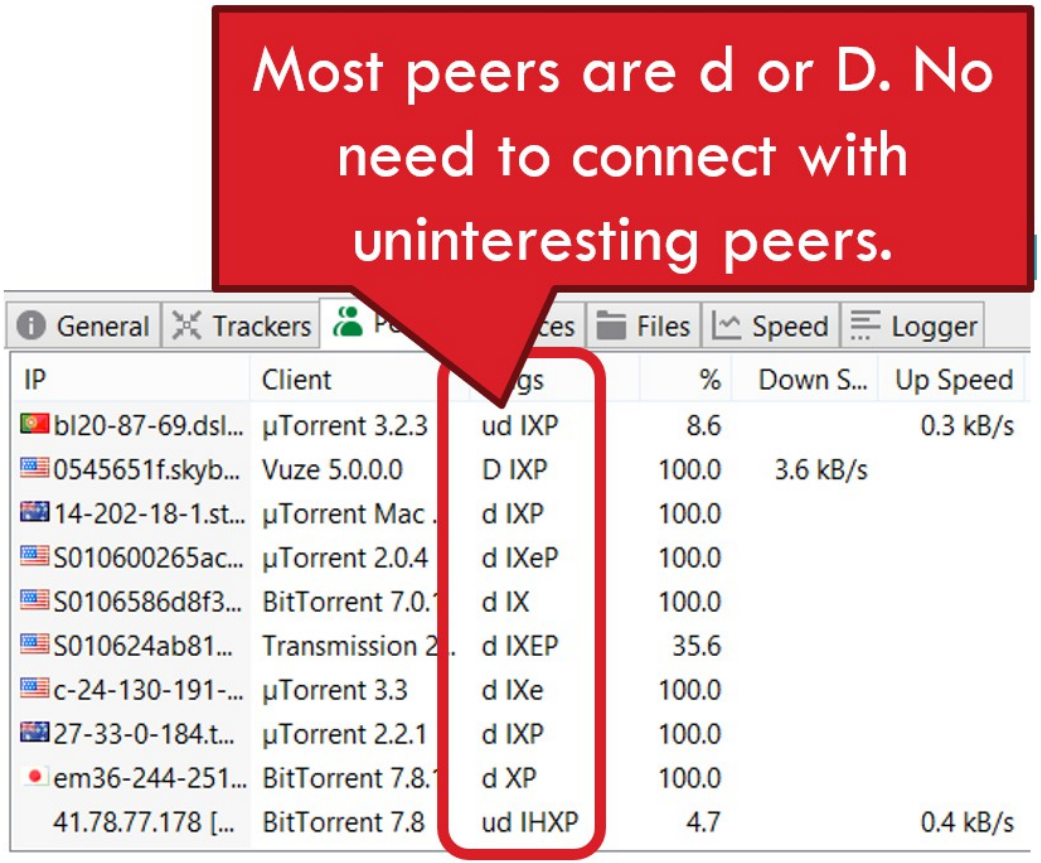
- d – interested and choked
- D – interested and unchoked
- K – uninterested and unchoked
- S – snubbed (no data received in 60 seconds)
- F – piece(s) failed to hash



BITTORRENT: STATO DELLE CONNESSIONI

Download control

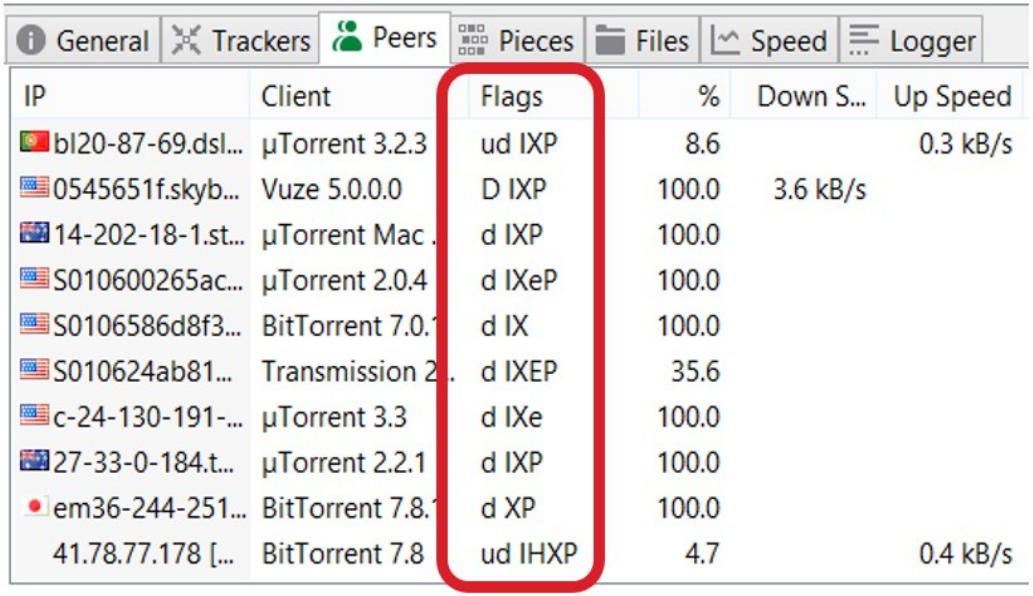
- d – interested and choked
- D – interested and unchoked
- K – uninterested and unchoked
- S – snubbed (no data received in 60 seconds)
- F – piece(s) failed to hash



BITTORRENT: STATO DELLE CONNESSIONI

Error states. Connection should be closed.

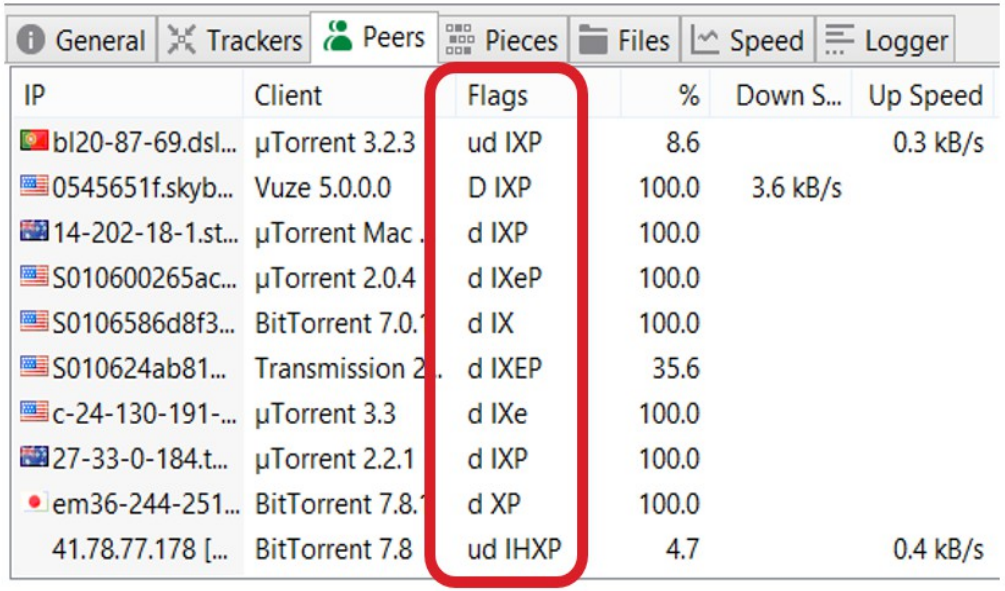
- rested and unchoked
- K – nterested and unchoked
- S – snubbed (no data received in 60 seconds)
- F – piece(s) failed to hash



IP	Client	Flags	%	Down S...	Up Speed
bl20-87-69.dsl...	µTorrent 3.2.3	ud IXP	8.6		0.3 kB/s
0545651f.skyb...	Vuze 5.0.0.0	D IXP	100.0	3.6 kB/s	
14-202-18-1.st...	µTorrent Mac .	d IXP	100.0		
S010600265ac...	µTorrent 2.0.4	d IXeP	100.0		
S0106586d8f3...	BitTorrent 7.0.1	d IX	100.0		
S010624ab81...	Transmission 2.	d IXEP	35.6		
c-24-130-191-...	µTorrent 3.3	d IXe	100.0		
27-33-0-184.t...	µTorrent 2.2.1	d IXP	100.0		
em36-244-251...	BitTorrent 7.8.1	d XP	100.0		
41.78.77.178 [...	BitTorrent 7.8	ud IHXP	4.7		0.4 kB/s

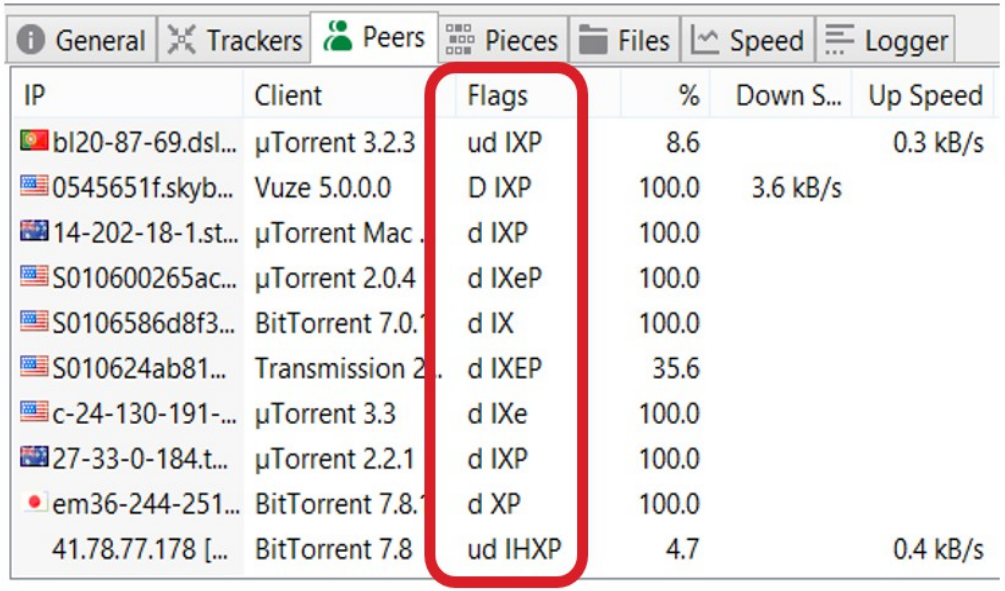
BITTORRENT: STATO DELLE CONNESSIONI

- Download control
  - d – interested and choked
  - D – interested and unchoked
  - K – uninterested and unchoked
  - S – snubbed (no data received in 60 seconds)
  - F – piece(s) failed to hash
- Upload control
  - u – interested and choked
  - U – interested and unchoked
  - O – optimistic unchoke
  - ? – uninterested and unchoked



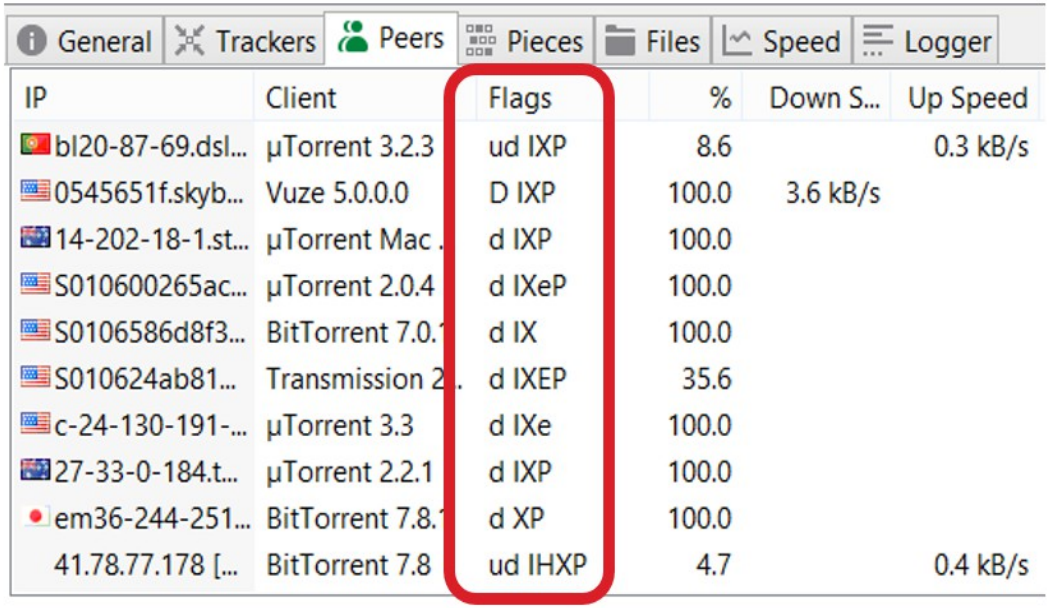
BITTORRENT: STATO DELLE CONNESSIONI

- Download control
  - d – interested and choked
  - D – interested and unchoked
  - K – uninterested and unchoked
  - S – snubbed (no data received in 60 seconds)
  - F – piece(s) failed to hash
- Upload control
  - u – interested and choked
  - U – interested and unchoked
  - O – optimistic unchoke
  - ? – uninterested and unchoked



BITTORRENT: STATO DELLE CONNESSIONI

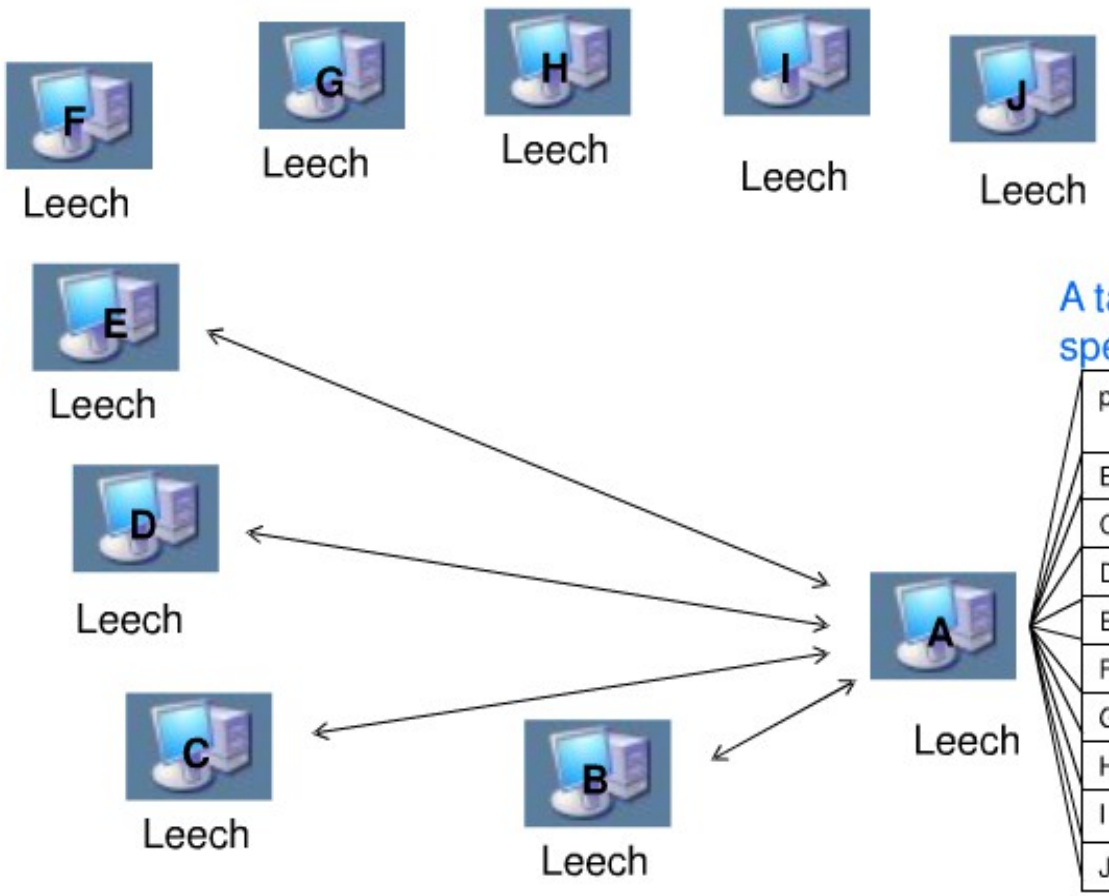
- Download control
  - d – interested and choked
  - D – interested and unchoked
  - K – uninterested and unchoked
  - S – snubbed (no data received in 60 seconds)
  - F – piece(s) failed to hash
- Upload control
  - u – interested and choked
  - U – interested and unchoked
  - O – optimistic unchoke
  - ? – uninterested and unchoked
- Connection information
  - I – incoming connection
  - E/e – Using protocol encryption



  - h – used UDP hole punching
  - P – connection uses µTP
- How was this peer located?
  - H – DHT (distributed hash table)
  - L – local peer discovery (multicast)
  - X – peer exchange

CHOCKING ALGORITHM: UN ESEMPIO

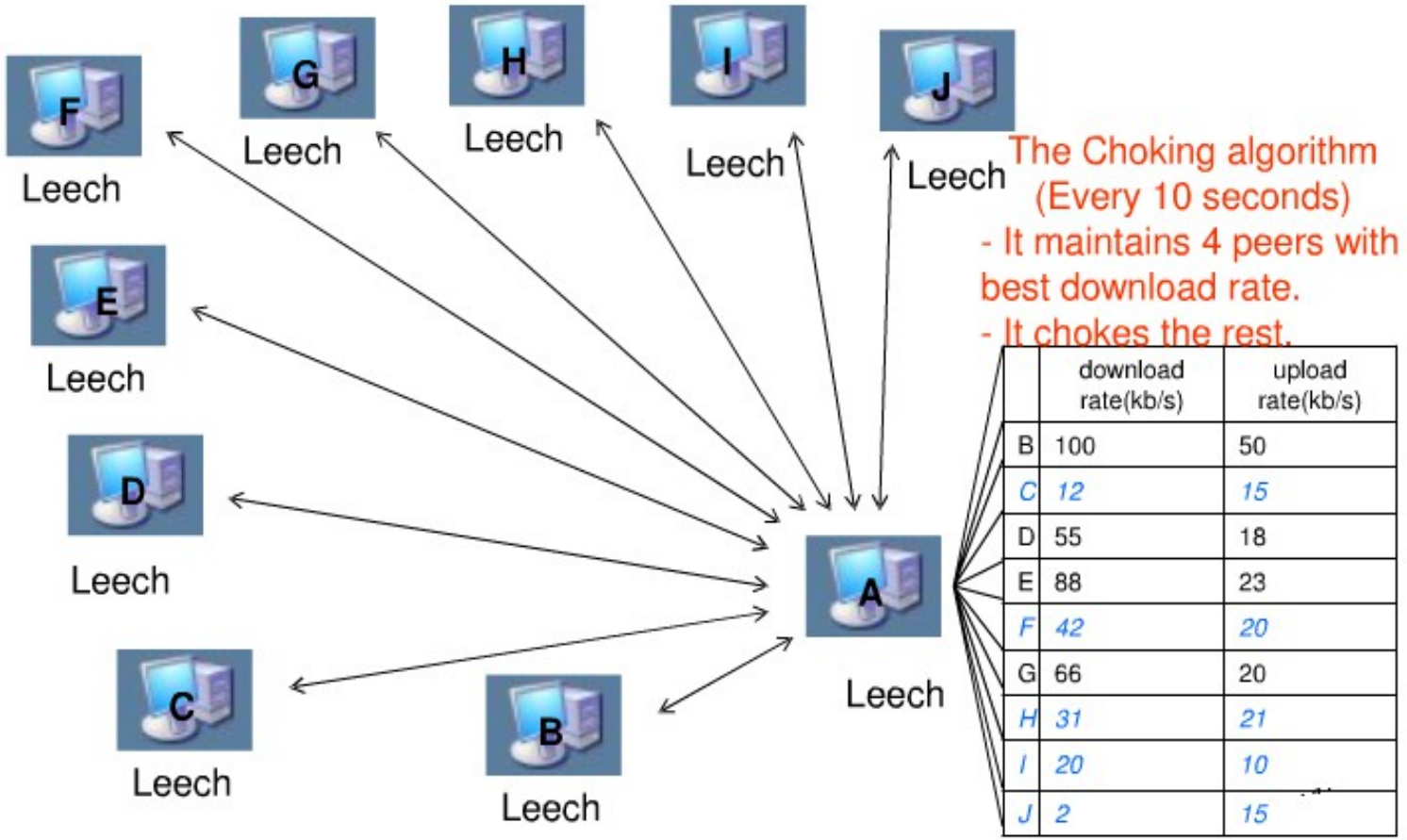
Taking A as an example:



A table maintaining the speed between peers

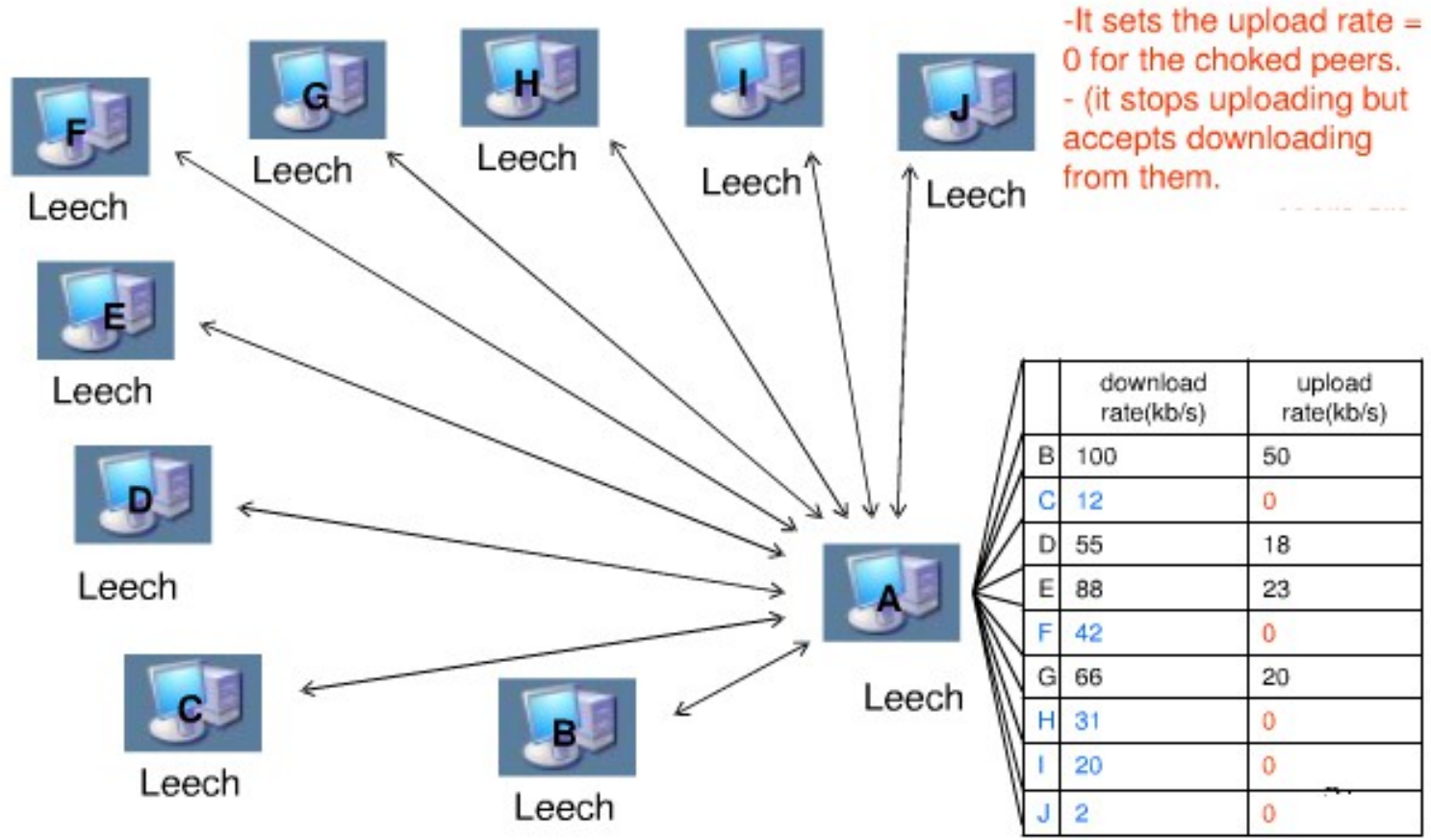
peer	download rate(kb/s)	upload rate(kb/s)
B	100	50
C	12	15
D	55	18
E	88	23
F	42	20
G	66	20
H	31	21
I	20	10
J	2	15

CHOCKING ALGORITHM: UN ESEMPIO



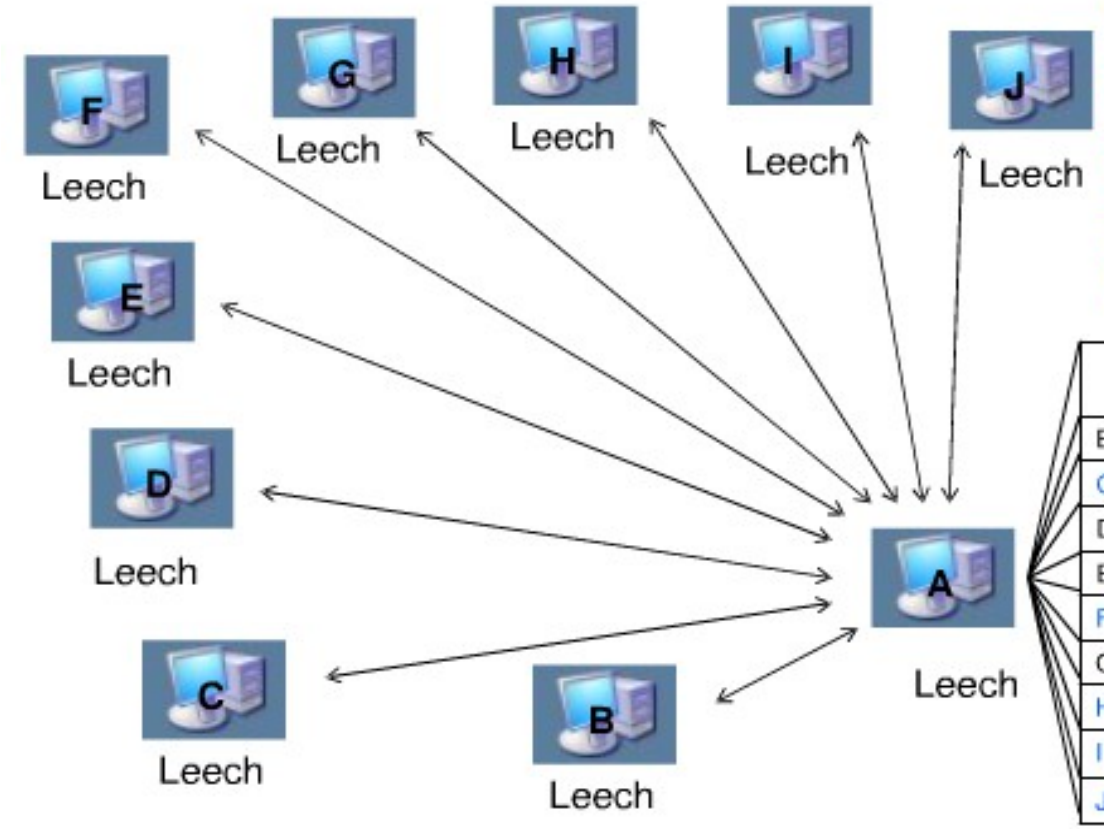
	download rate(kb/s)	upload rate(kb/s)
B	100	50
C	12	15
D	55	18
E	88	23
F	42	20
G	66	20
H	31	21
I	20	10
J	2	15

CHOCKING ALGORITHM: UN ESEMPIO



	download rate(kb/s)	upload rate(kb/s)
B	100	50
C	12	0
D	55	18
E	88	23
F	42	0
G	66	20
H	31	0
I	20	0
J	2	0

CHOCKING ALGORITHM: UN ESEMPIO

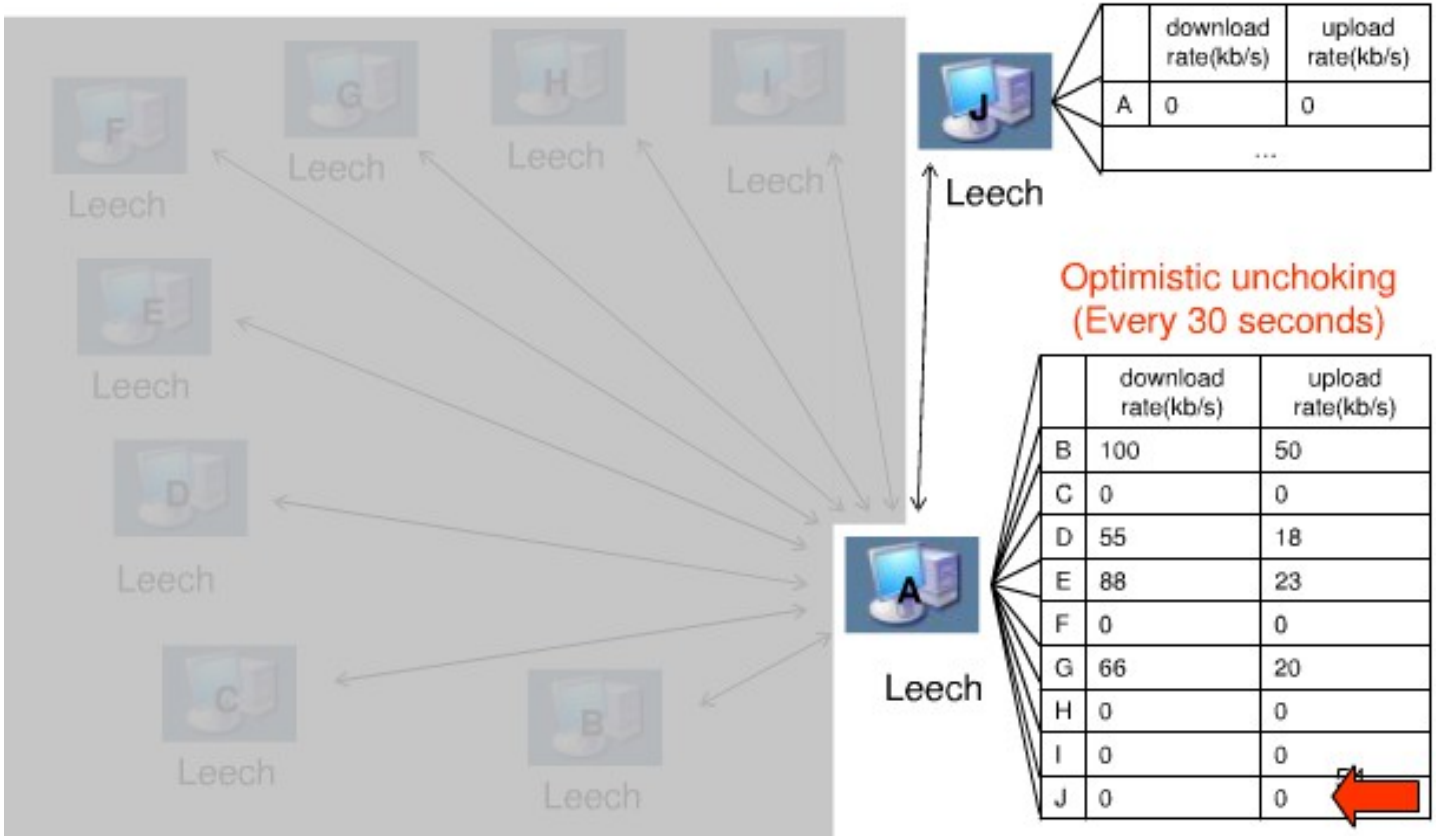


-As other peers are also running the choking algorithm.
-Setting upload rate =0 causes the corresponding peers set the upload rate to A = 0.
-It makes the download rate in A's table become 0.

	download rate(kb/s)	upload rate(kb/s)
B	100	50
C	0	0
D	55	18
E	88	23
F	0	0
G	66	20
H	0	0
I	0	0
J	0	0

CHOCKING ALGORITHM: UN ESEMPIO

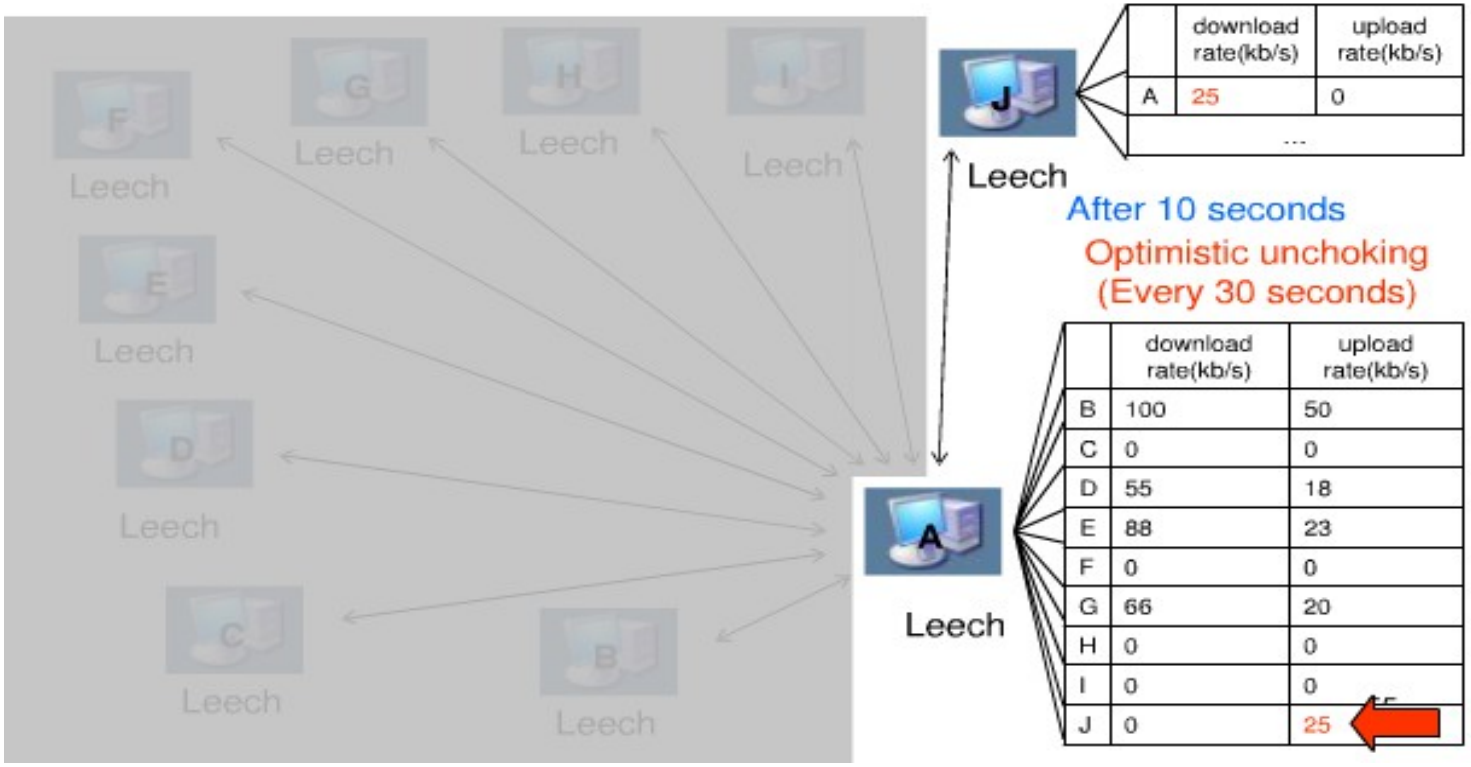
Randomly choose 1 choked peer and unchoke it. (i.e., will upload data to it)



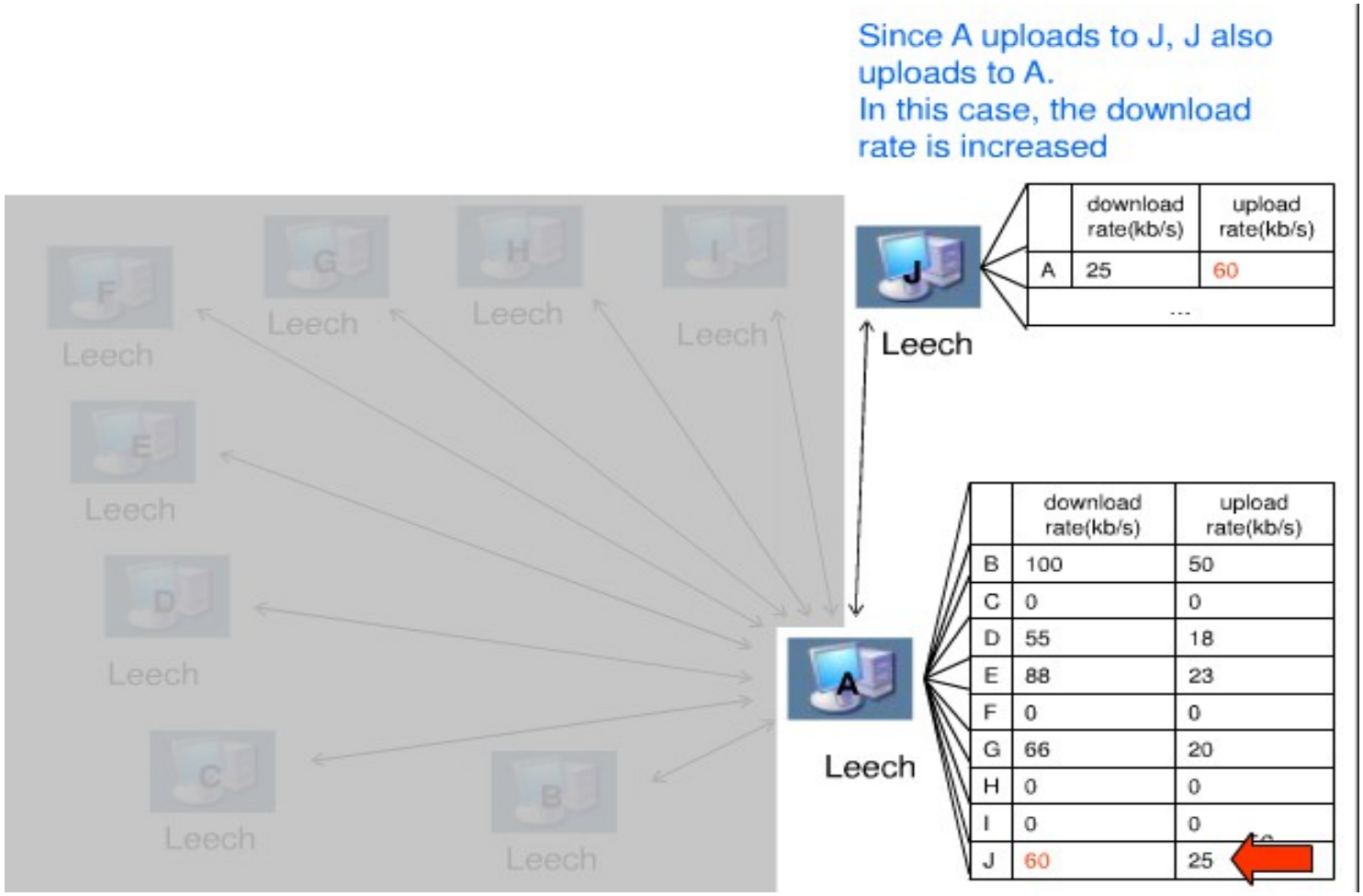
	download rate(kb/s)	upload rate(kb/s)
A	0	0
...		

	download rate(kb/s)	upload rate(kb/s)
B	100	50
C	0	0
D	55	18
E	88	23
F	0	0
G	66	20
H	0	0
I	0	0
J	0	0

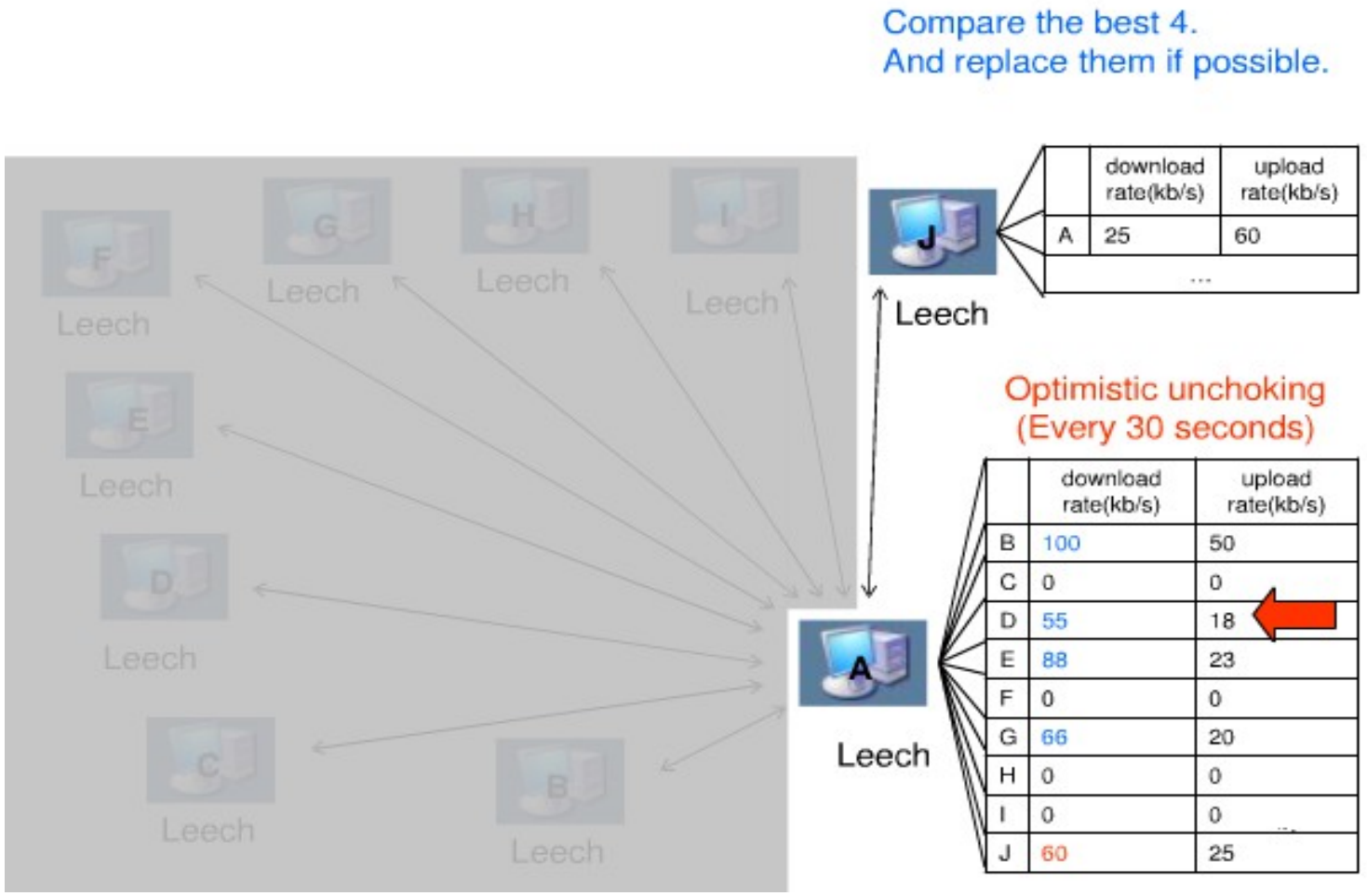
CHOCKING ALGORITHM: UN ESEMPIO



CHOCKING ALGORITHM: UN ESEMPIO



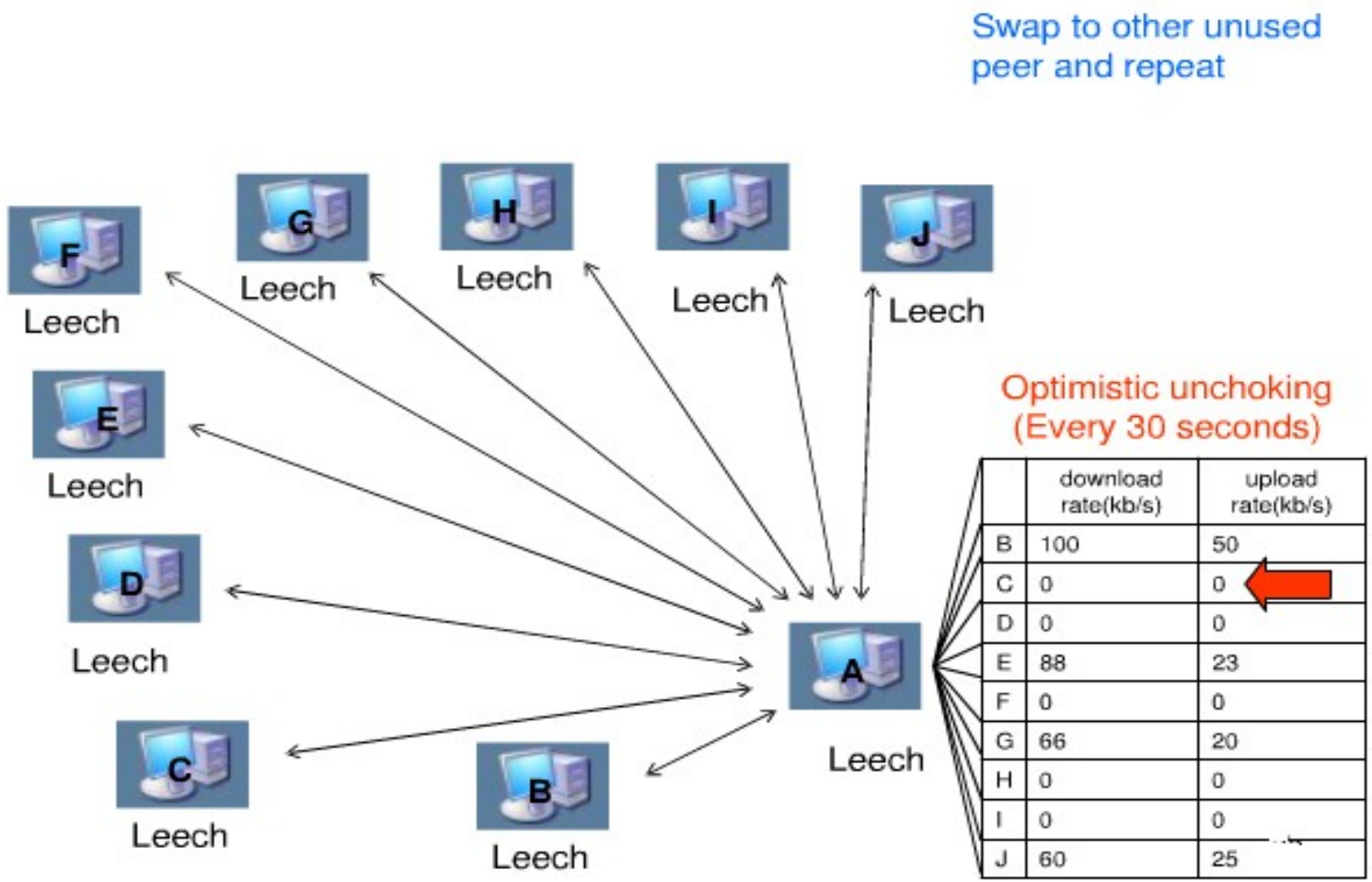
CHOCKING ALGORITHM: UN ESEMPIO



	download rate(kb/s)	upload rate(kb/s)
A	25	60
...		

	download rate(kb/s)	upload rate(kb/s)
B	100	50
C	0	0
D	55	18
E	88	23
F	0	0
G	66	20
H	0	0
I	0	0
J	60	25

CHOCKING ALGORITHM: UN ESEMPIO



	download rate(kb/s)	upload rate(kb/s)
B	100	50
C	0	0
D	0	0
E	88	23
F	0	0
G	66	20
H	0	0
I	0	0
J	60	25

CHOCKING: CLUSTERING DEI PEER

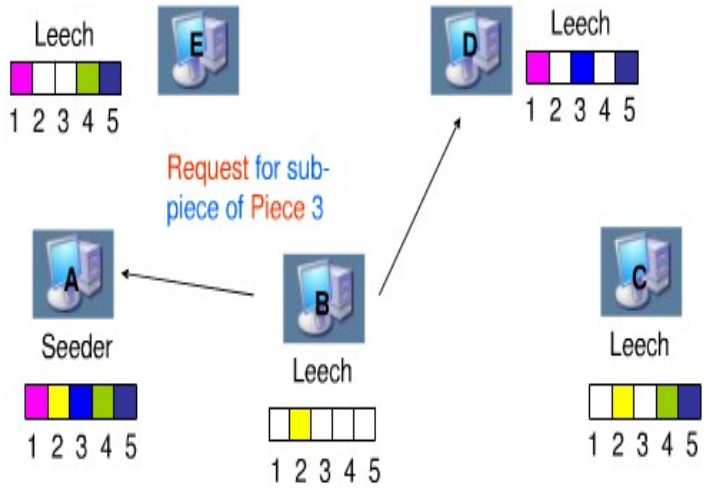
- supponiamo che P esegua un optimistic unchoking verso Q. P può diventare uno degli "uploader preferiti" di Q:
  - Q inizia a inviare dati a P
  - se Q invia dati con frequenza adeguata, Q può diventare, a sua volta, uno degli "uploader preferiti" di P.
- se i due parner sono soddisfatti dalla velocità reciproca di scambio dei dati:
  - si inseriscono a vicenda nella lista dei "rispettivi uploader"
  - continuano a scambiardi dati fino a che uno dei due non trova un partner migliore
- i peers che possono scambiare i dati a frequenza simile, tendono a "trovarsi uno con l'altro" e scambiano i dati
  - se un peer P vuole scaricare dati da un altro peer ad una certa velocità. P deve offrire dati con una velocità confrontabile
  - se un peer P non offre dati, viene "snobbato" da tutti gli altri peer e rimane isolato

SCELTA DEI PEZZI

- Strict Priority
  - Complatare l'"assemblaggio" di un pezzo prima di chiederne un altro
- Rarest First
  - Scaricare i prezzi più rari per primi
- Random First Piece
  - Scegliere un peer random nella fase di bootstrap
- Endgame
  - Chiedere a tutti i blocchi rimanenti quando il download sta per finire

STRICT PRIORITY

- Una volta che si è chiesto il blocco di un pezzo,
  - richiedere tutti gli altri blocchi dello stesso pezzo prima di richiedere un blocco di un qualsiasi altro pezzo
- Idea
  - è importante scaricare un pezzo il prima possibili
    - solo dei pezzi completi possono essere negoziati
  - l'algoritmo di chocking favorisce i peer che hanno pezzi interi da scambiare con i partner



request: <len=0013><id=6><index><begin><length>
The request message is fixed length, and is used to request a block. The payload contains the following information:

index: integer specifying the zero-based piece index
begin: integer specifying the zero-based byte offset within the piece
length: integer specifying the requested length.

Strict priority

Once this request has been made, it only asks for all sub-pieces in pieces 3 first. (use at most 5 connections)

It will not ask for other pieces until it has finished the download of piece 3.

RANDOM FIRST PIECE

- inizialmente, un leacher non possiede alcun pezzo e non può offrire nessun contenuto agli altri peer dello swarm
  - è importante che un peer acquisisca un pezzo il prima possibile, per iniziare la negoziazione dei pezzi con gli altri peer, per non essere "strozzato" da tutti i peer della comunità
- i pezzi sono selezionati in modo casuale, per i primi 4 pezzi scaricati
- scopo: è possibile che il download dei prezzi più rari sia più lento
  - in particolare, se i pezzi sono presenti su un solo peer
- un peer che non possiede alcun pezzo non può cooperare con nessuno
  - deve aspettare un optimistic unchoke
  - il primo pezzo spesso viene ricevuto da altri peer che eseguono optimistic unchocking
  - "fast starting", dopo di cui si passa alla politica di rarest first

RANDOM FIRST PIECE

Supponiamo che all'inizio un peer riceva il suo primo pezzo mediante un Optimistic Unchocking:

- con un avalocità di upload di 20kB/s, ogni unchoked peer riceve i pezzi a 5kB/s (4 in parallelo)
- un pezzo di 256kB, richiede 51 a 5kB/s per essere scaricato
- ma...., un Optimistic Unchocking dura solamente 30s
- un optimistic unchocking può non diventare un regular unchocking quando il peer non possiede pezzi da inviare
- è più veloce completare l'ultimo blocco del pezzo da un altro peer che effettua un Optimistic Unchocking se il peer non è tra quelli più rari

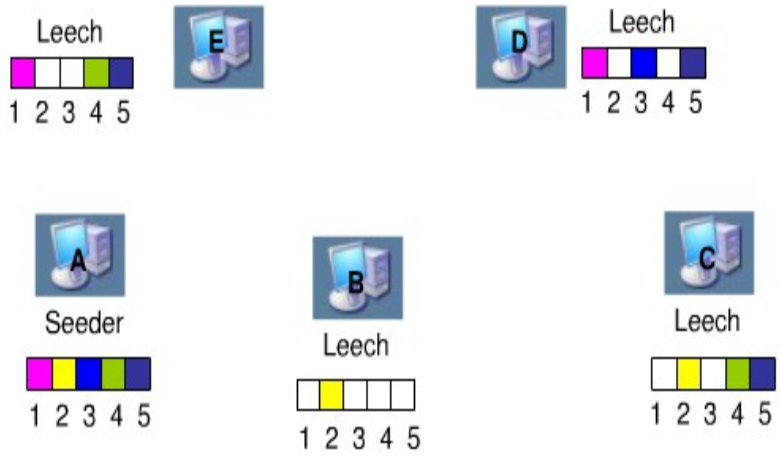
SCELTA DEI PEERS: LOCAL RAREST FIRST

Scopo del rarest piece first: massimizzare l'entropia dei pezzi all'interno dei .torrent

- se il seeder di un file si disconnette dallo swarm, è possibile che un pezzo raro non sia più disponibile nella rete.
  - questo renderebbe l'assemblaggio di un file impossibile.
- se esistono pochi seeder del file nella rete, con una capacità di upload limitata, questa strategia garantisce che il file sia scaricato da più downloader, diminuendo anche il carico sul seeder.
- un peer che acquisisce un pezzo raro, ottiene un pezzo che probabilmente verrà scelto da diversi peer per il download.
  - in questo modo, è probabile che il peer sia inserito nella liste di upload di diversi nodi

LOCAL RAREST FIRST

- Ogni peer conosce i pezzi posseduti dai peer nel suo Peer Set e può calcolare la disponibilità di ogni pezzo.
- Rarest Piece Set: l'insieme di pezzi con disponibilità minima
  - aggiornato quando un peer riceve un messaggio Have o Bitfield
- Local Rarest First: seleziona un peer nel Rarest Piece Set.
  - replica i rarest pieces il più velocemente possibile
  - reperire un "pezzo prezioso" da scambiare
  - download di pezzi unici dai seeder.

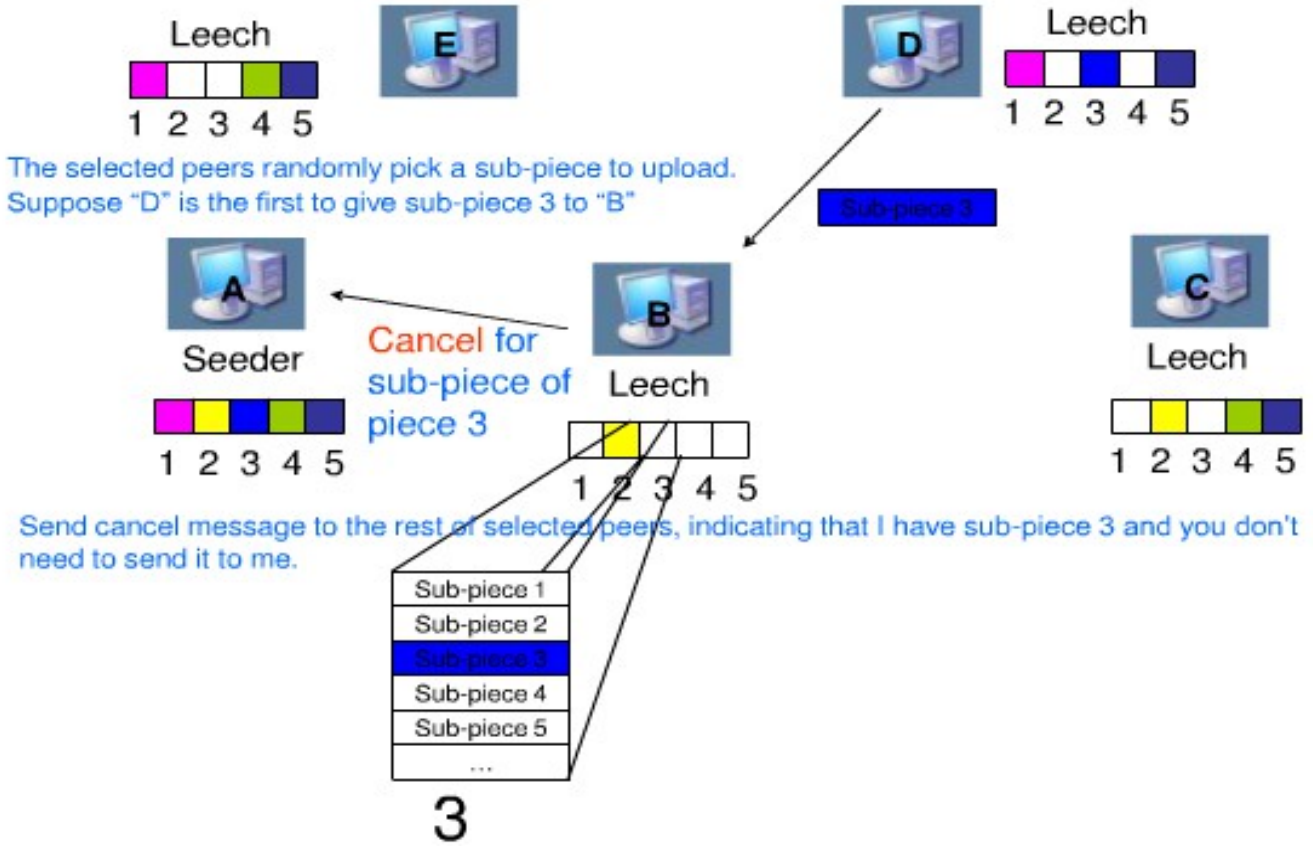


Rarest First Piece

Since both A and D have Piece 3, **both A and D are selected**

piece	Total number in the swarm of B	Corresponding peers
1	3	(A, D, E)
2	3	(A, C)
3	2	(A, D)
4	3	(A, C, E)
5	4	(A, C, D, E)

LOCAL RAREST FIRST

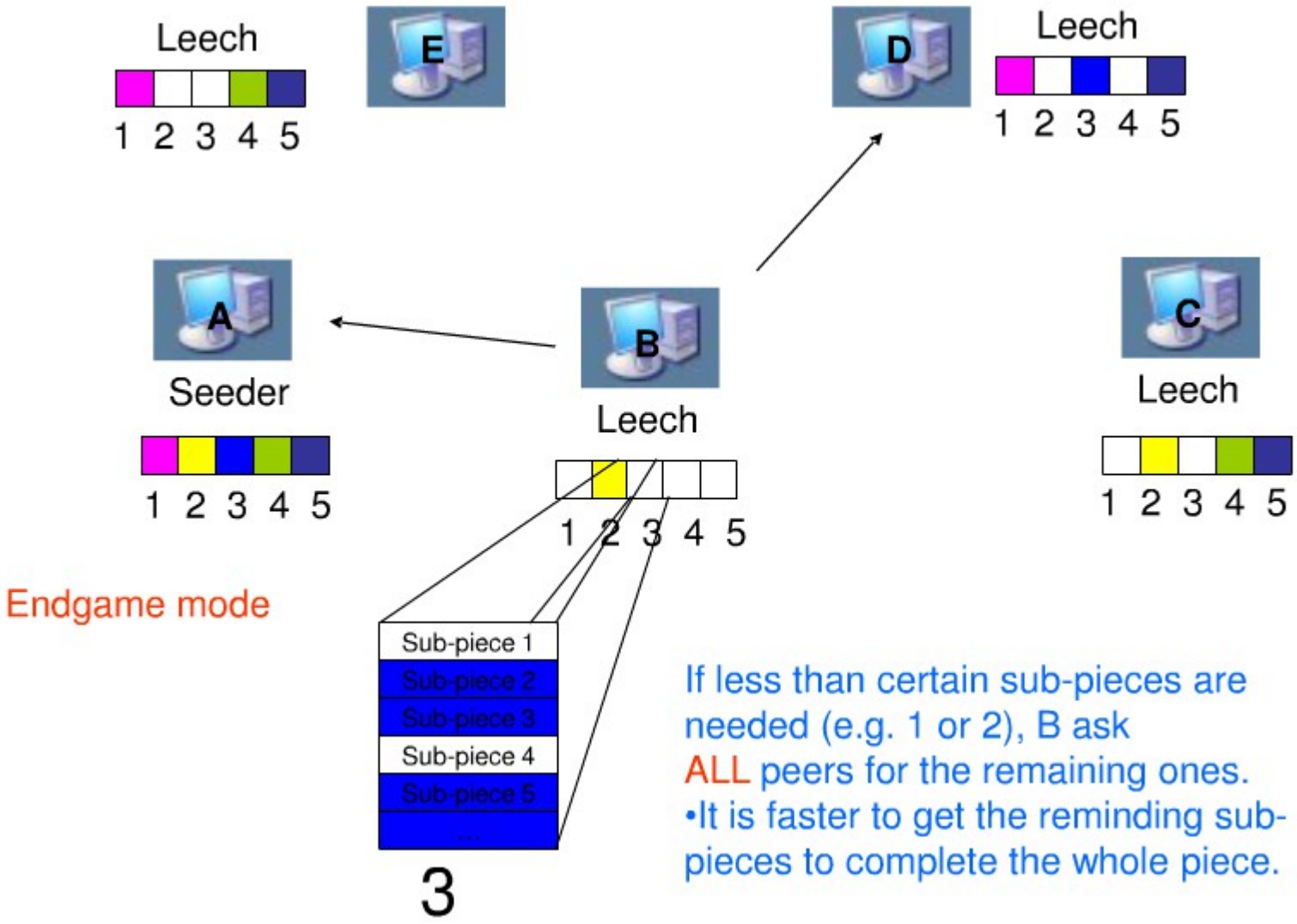


ENDGAME

- **Osservazione:** i downlaod rallentano quando ci si avvicina alla fine dello scaricamento del file
  - i peer caratterizzati da un basso livello di banda rallentano il trasferimento verso la fine dello scaricamento, perchè il dowload da questi peer non può essere sovrapposto al download da altri peer

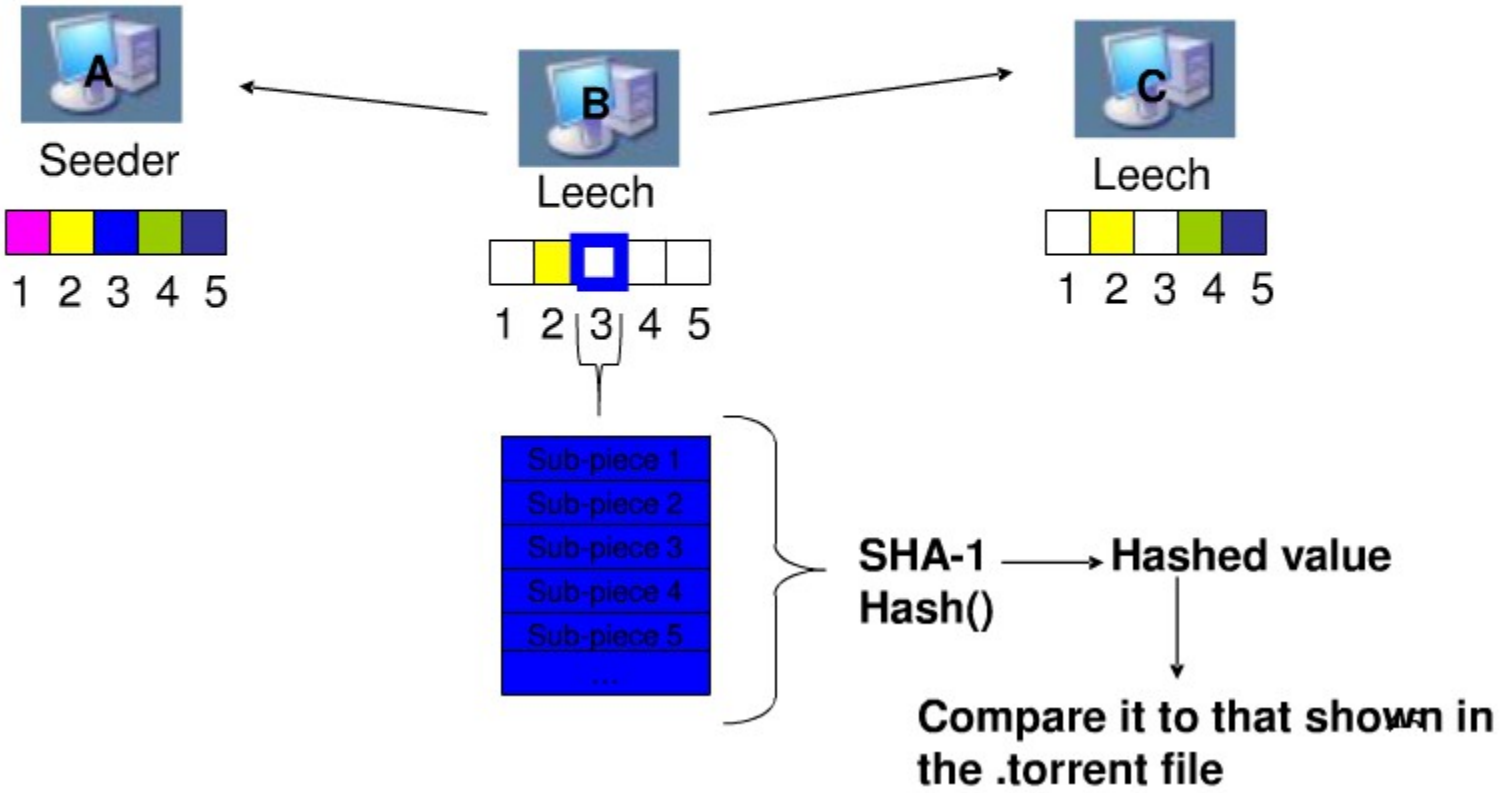
- **End Game Policy**
  - gli ultimi pezzi del file sono richiesti a tutti i peer che possiedono quei pezzi del file
  - per evitare lo spreco di banda, quando il pezzo è ricevuto dal peer che lo ha richiesto, tutti I download iniziati in parallelo vengono cancellati
  - solo un piccolo spreco di banda, perchè il download parallelo viene eseguito solo per un breve periodo.

ENDGAME


Leech
1 2 3 4 5
E
D
Leech
1 2 3 4 5
A
Seeder
1 2 3 4 5
B
Leech
1 2 3 4 5
C
Leech
1 2 3 4 5
Endgame mode
Sub-piece 1
Sub-piece 2
Sub-piece 3
Sub-piece 4
Sub-piece 5
...
3
If less than certain sub-pieces are needed (e.g. 1 or 2), B ask ALL peers for the remaining ones.
•It is faster to get the reminding sub-pieces to complete the whole piece.

DATA INTEGRITY CHECK

Calculate the SHA-1 hash code for this just downloaded piece.
1. If the code matches with the one shown in the torrent file, it announce to all active peers it has this piece.
2. If not, it drops that piece and download again.



STRATEGIA TIT FOR TAT: CARATTERISTICHE

- Cooperative: coopera sempre al primo turno
  - in BitTorrent, questo corrisponde a fare l'upload del contenuto, come prima mossa (optimistic move)
  - cooperazione: fornire banda alla comunità
- Reciprocity: interrompere la collaborazione se il partner non collabora
  - in BitTorrent: "chocke" dell'upload del contenuto se il partner non invia i dati o invia i dati a bassa frequenza
  - defezione: choke l' upload verso un peer che non collabora
- Forgiveness:
  - riprendere la collaborazione se il partner è disposto a collaborare, anche se non ha collaborato nel passato
  - in BitTorrent a peer riprende l'upload dei dati quando il partner collabora
- Not envious: non si cerca di far meglio del parner

BITTORRENT AND TIT FOR TAT

- Cooperative: upload verso qualche peer
- Change strategy: se un peer non invia contenuti, chocke
- Optimistic Unchoke: dimenticati del comportamento passato, e dai ad un eer l'opportunità di cooperare di nuovo
  - Prima mossa del dilamma del prigioniero: prova a collaborare per veere se si stabilisce la reciprocità
  - Permette il bootstrap di nuovi peers
    - evaluate the upload capacity of new peers
    - a peer which has just entered into the overlay does not own any piece, therefore it cannot upload content to any other peer
    - Provide new peers with their first piece as soon as possible
      - so that they can start to exchange pieces

BITTORRENT VERSIONS

- All results presented are from mainline client 4.0.2
- Mainline is the reference BitTorrent implementation
  - written in Python
  - developed by Bram Cohen, still maintained by his company
- Mainline 4.0.2 was released in may 2005
- Major modifications since 2005
  - Tracker less extension: use of Kademlia. To be seen in the next lessons
  - GUI improvement
  - Localization
  - UpnP
  - Etc.
- No major modification to the core algorithms (peer and piece selection)

CONCLUSIONI

- Swarm Discovery
  - search engine
- Peer Discovery
  - Central Tracker, Distributed Tracker (DHT), PEX protocol
- Data Discovery
  - Information Exchange between the peers
- Peer Selection
  - choking algorithms
- Piece Selection
  - Strict Priority, Rarest First, Random First Piece, Endgame

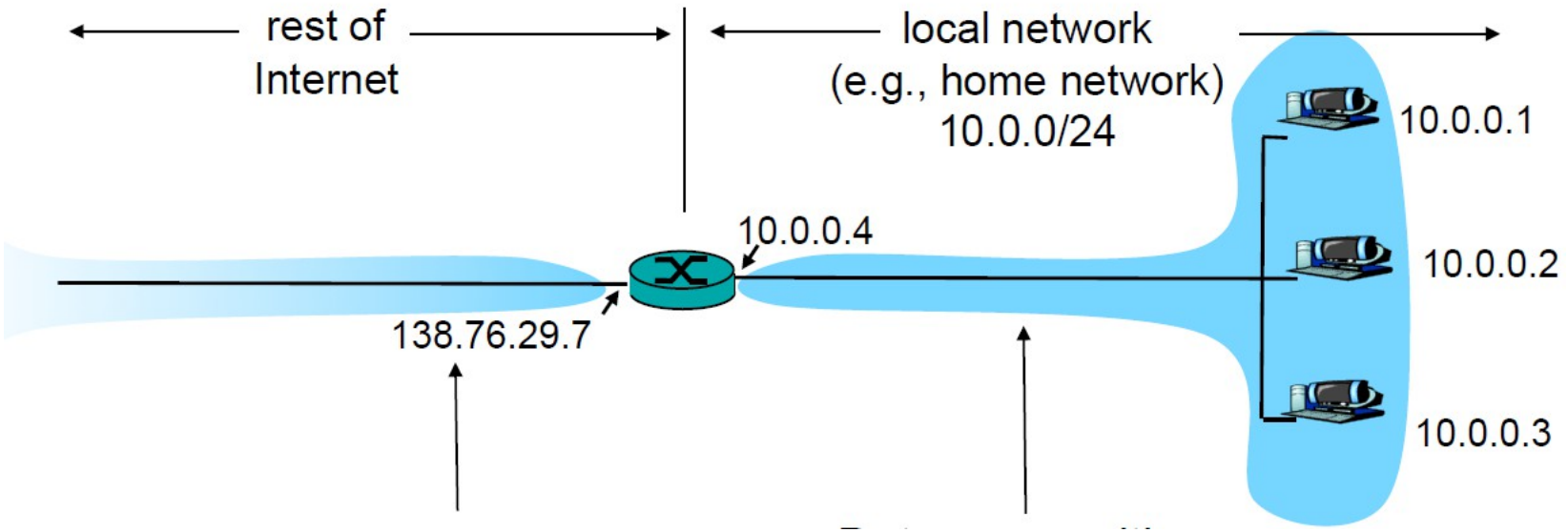
NETWORK ADDRESS TRANSLATION

- Problema: scarsezza di indirizzi IPv4
  - esplosione del numero di device
  - solo 32 bit a disposizione
- idea: la rete locale usa solo un indirizzo IP per il mondo esterno
  - non serve un range di indirizzi IP per un ISP: solo un indirizzo IP per tutti i nodi serviti
  - possibilità di cambiare l'indirizzo di un host senza notificarlo al mondo esterno
  - possibilità di cambiare l'indirizzo dell'ISP senza cambiare gli indirizzi dei nodi nella rete locale
  - meccanismo di sicurezza: gli end host all'interno di una rete locale non sono indirizzabili direttamente, non visibili al mondo esterno
  - facilita l'amministazione della rete

NETWORK ADDRESS TRANSLATION

- NAT (Network Address Translation) e firewalls sono meccanismi utilizzati per controllare la comunicazione tra due diverse sottorete
- Firewalls:
  - aumentano la sicurezza degli host di una rete bloccando connessioni non autorizzate provenienti dall'esterno
- NAT
  - effettuano la conversione tra spazi di indirizzi diversi appartenenti a sottoreti diverse

NAT: SCHEMA DI FUNZIONAMENTO



Tutti i datagrammi che escono dalla rete hanno lo stesso indirizzo NAT 138.76.29.7, ma diversi numeri di porta

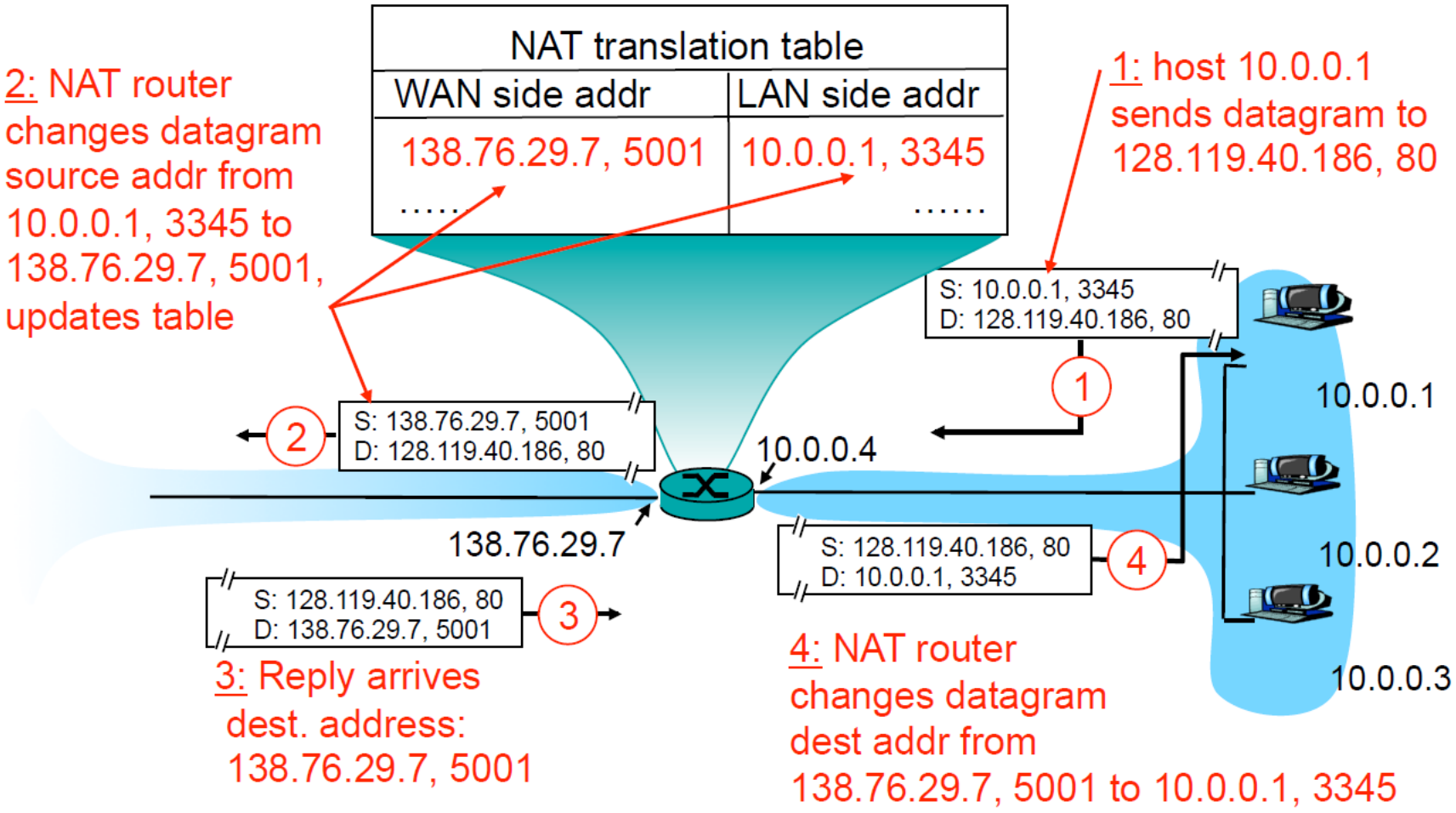
Tutti i datagrammi con sorgente/destinazione in questa sottorete hanno indirizzo 10.0.0/24 come sorgente/destinazione

NAT: IMPLEMENTAZIONE

Funzioni del NAT:

- outgoing datagram: sostituire in ogni datagram uscente la coppia (IP sorgente, #porta) con (NAT IP, #nuovaporta) dove #nuovaporta è un nuovo numero di porta generato dal NAT
  - i client/servet remoti rispondono usando (NAT IP, #nuova porta) come indirizzo destinazione
- registrare nella NAT Translation Table ogni coppia
  (IP sorgente, #porta) (NAT IP, #nuovaporta)
- incomig datagram: sostituire (NAT IP address, new port #) nel campo destinazione di ogni incomimg datagram con il corrispondente (IP sorgente, #porta) memorizzato nella tabella di traduzione
  - scartare i datagrammi che vengono ricevuti dal NAT se non esiste un'associazione nella tabella di traduzione per l'indirizzo ricevuto

NETWORK ADDRESS TRANSLATION



Nella tabella può essere registrato anche il destinatario, in modo da controllare che i pacchetti in ricezione arrivino da host contattati precedentemente

NETWORK ADDRESS TRANSLATION

- utilizza il numero di porta a 16-bit:
  - ~65000 connessioni simultanee con un singolo indirizzo di LAN
  - molto utile contro l'esaurimento degli indirizzi IP
- router che operano anche come NAT
- ...ma l'uso di NAT è molto controverso
  - i router dovrebbero occuparsi solo del livello di rete, invece manipolano le porte
  - deve essere preso inconsiderazione dali sviluppatori di applicazioni P2P
- la scarsezza di indirizzi IP dovrebbe essere invece risolta con IPv6
- gli indirizzi assegnati alle sottoreti interne appartengono ad una delle seguenti zone
  - 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16