Il protocollo BitTorrent

Felice Colucci

Argomenti

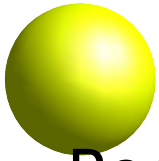
- Introduzione
- Come funziona: un breve sguardo
- I componenti
- Come creare un torrent
- Come funziona: qualche dettaglio
- Selezione dei chunck
- Algoritmi di choking
- Client
- Conclusioni

Che cosa è

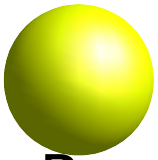
- Un protocollo peer-to-peer per il file sharing
- È il nome del client ufficiale
- Scritto da Bram Cohen
- Scritto in Python
- Pensato per file di grossa dimensione (ISO, dvd,...)

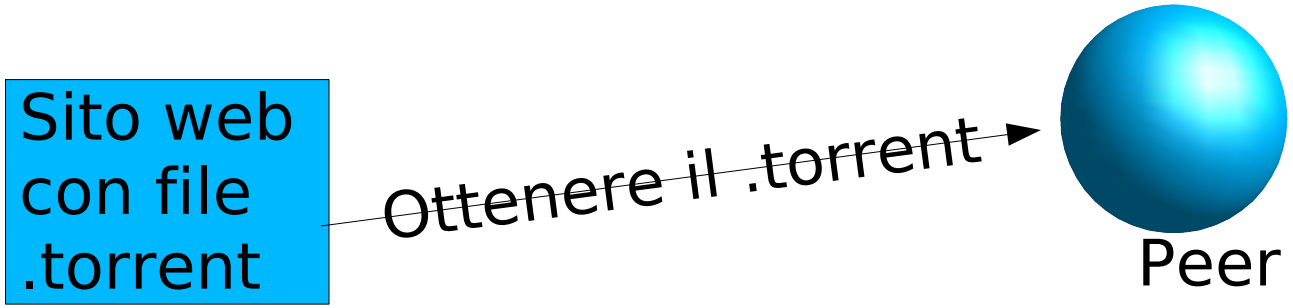
- Introduzione
- Come funziona: un breve sguardo
- I componenti
- Come creare un torrent
- Come funziona: qualche dettaglio
- Selezione dei chunck
- Algoritmi di choking
- Client
- Conclusioni

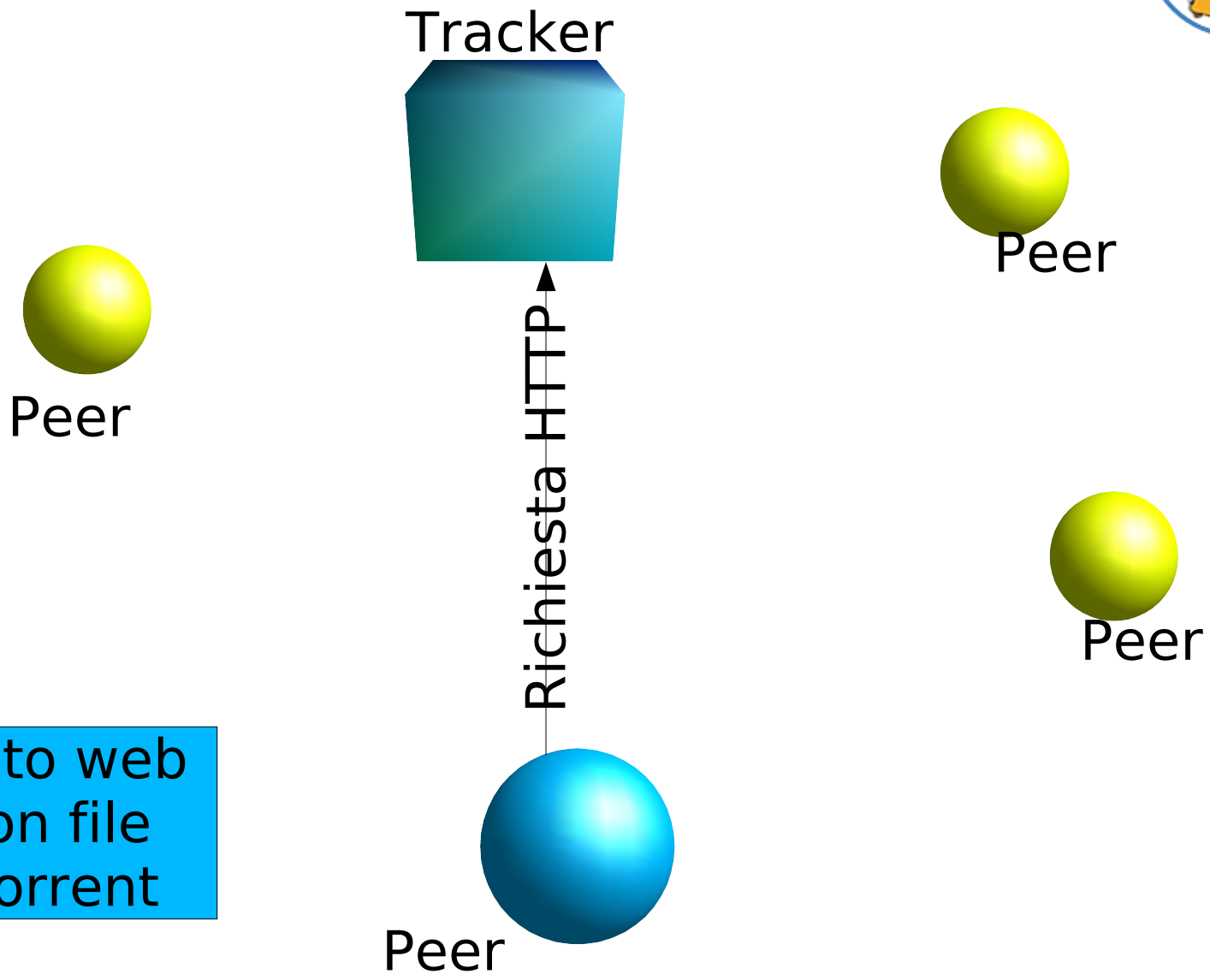

Tracker

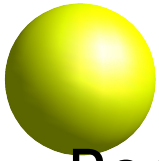

Peer


Peer

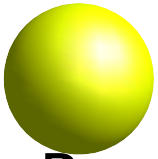

Peer


Sito web con file .torrent
Ottenere il .torrent
Peer

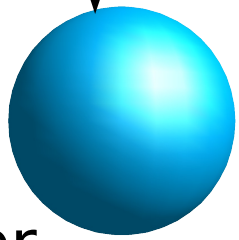

Tracker
Peer
Peer
Richiesta HTTP
Peer
Peer
Sito web con file .torrent

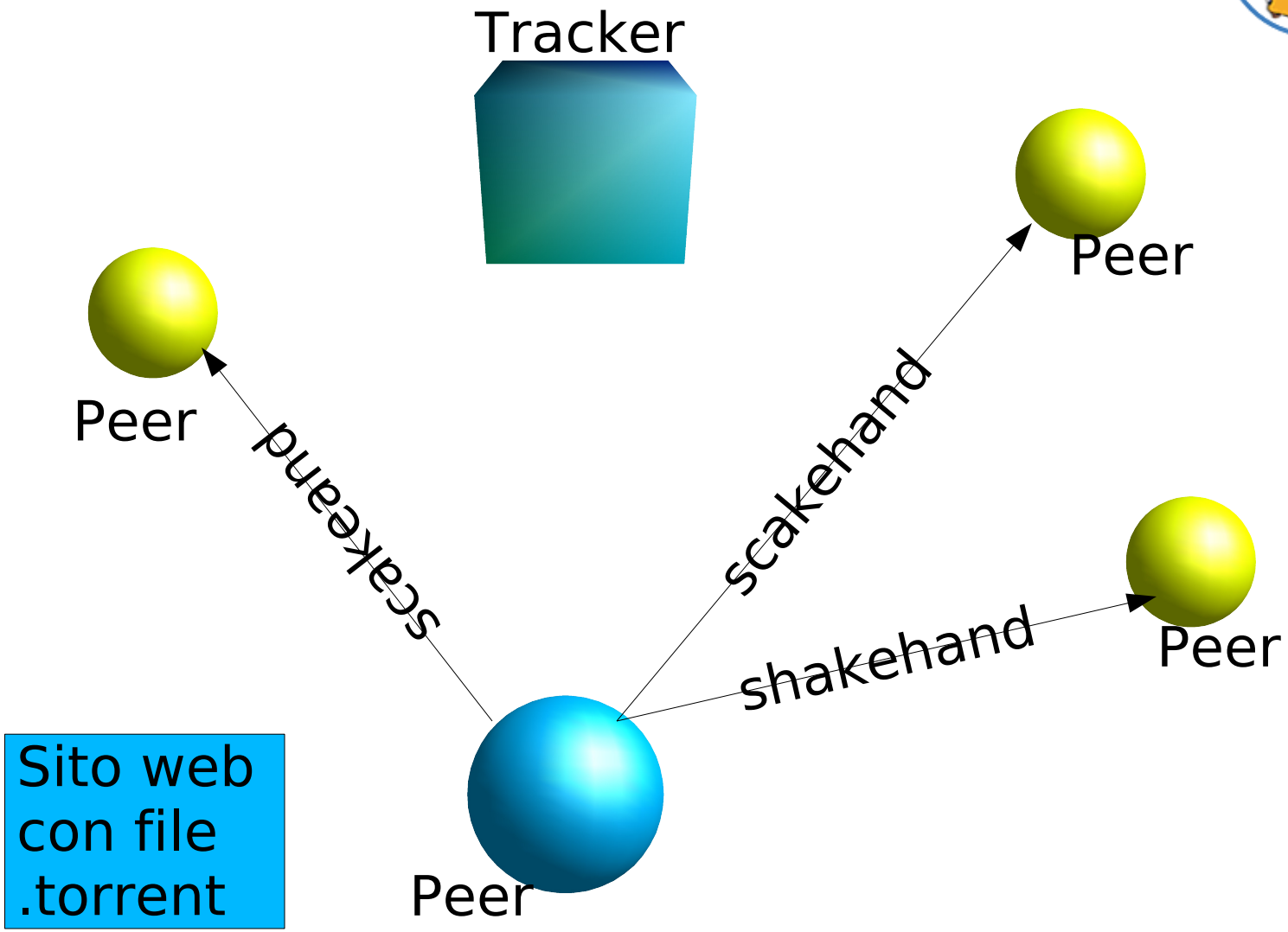


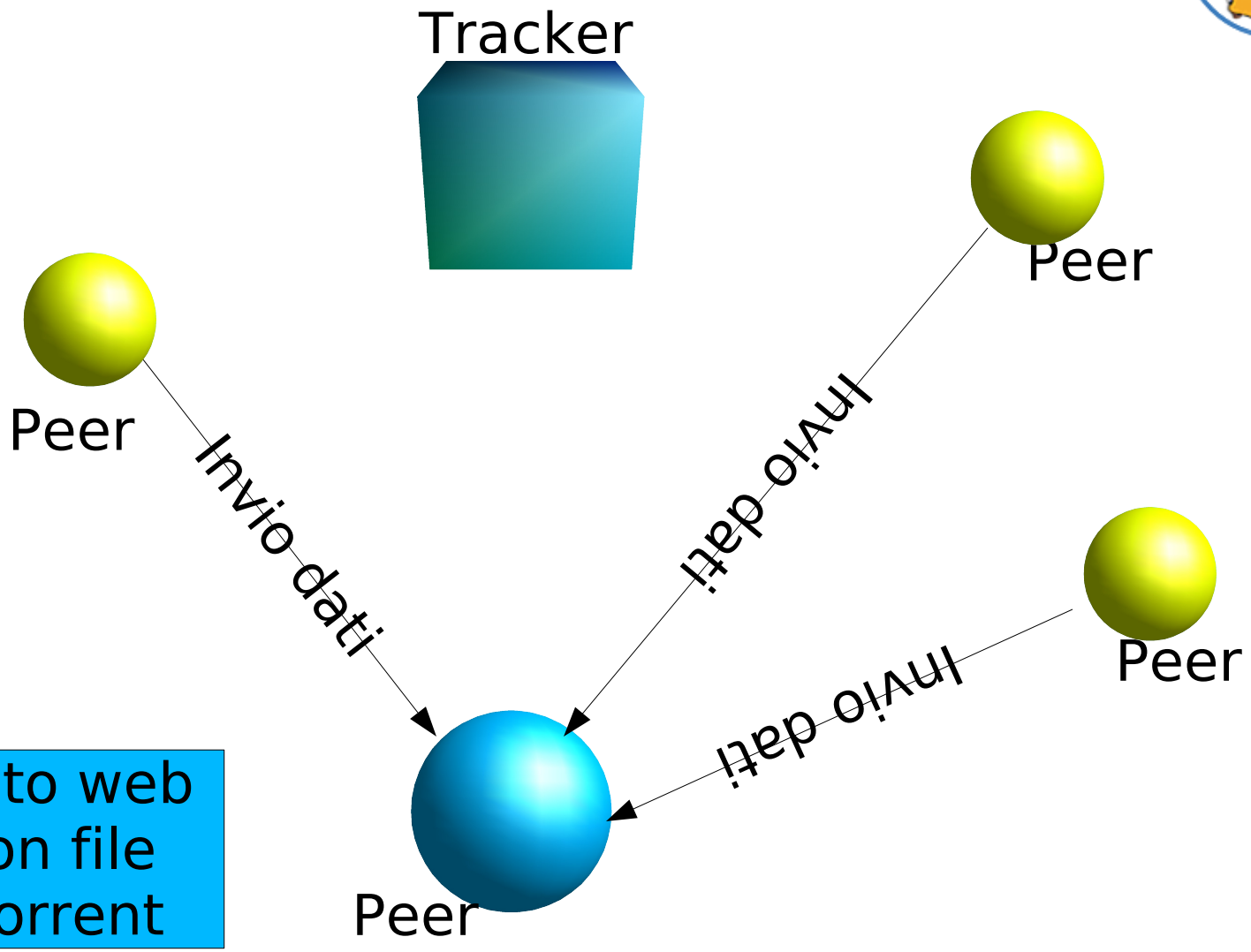
Lista di peer

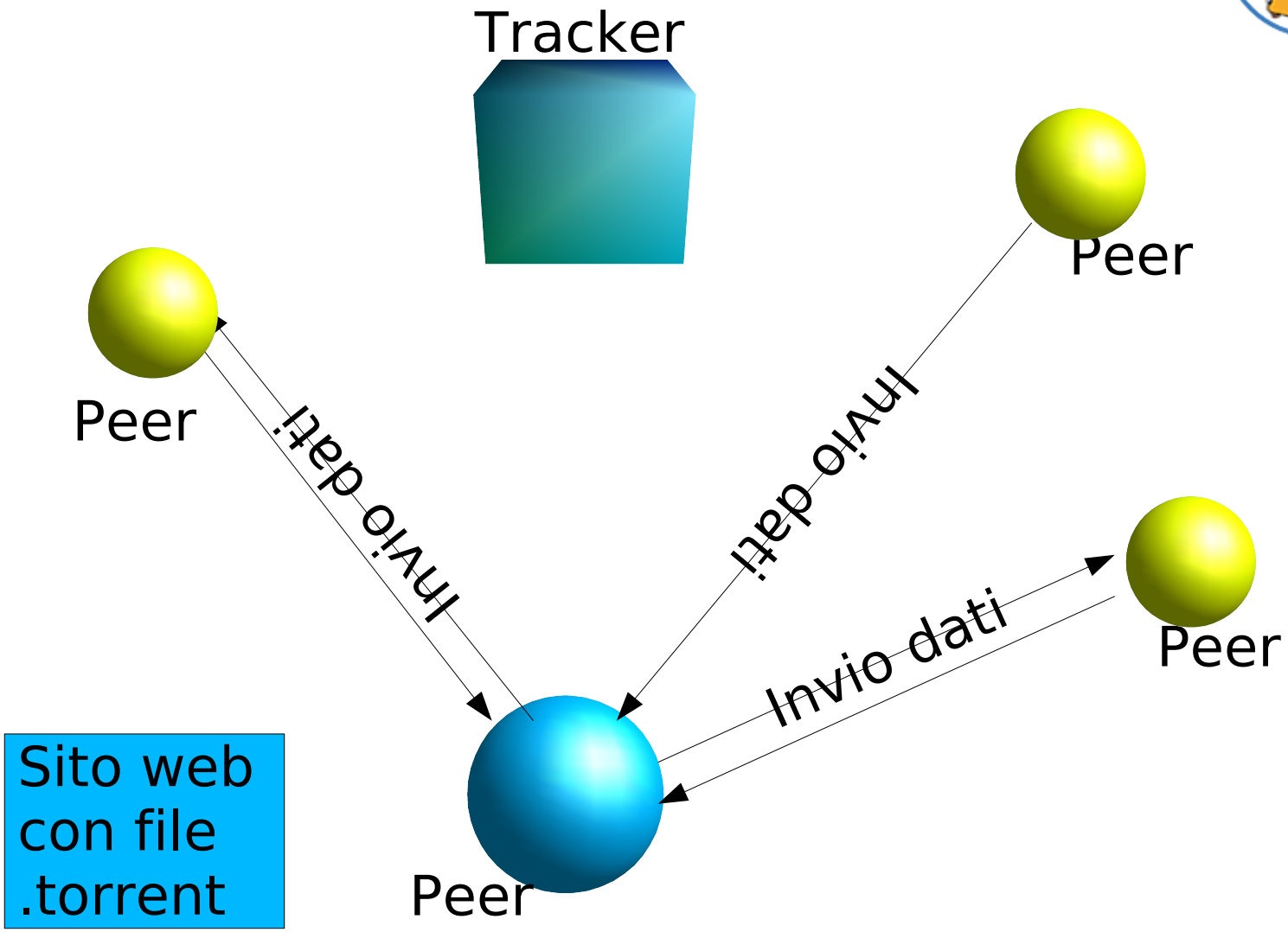


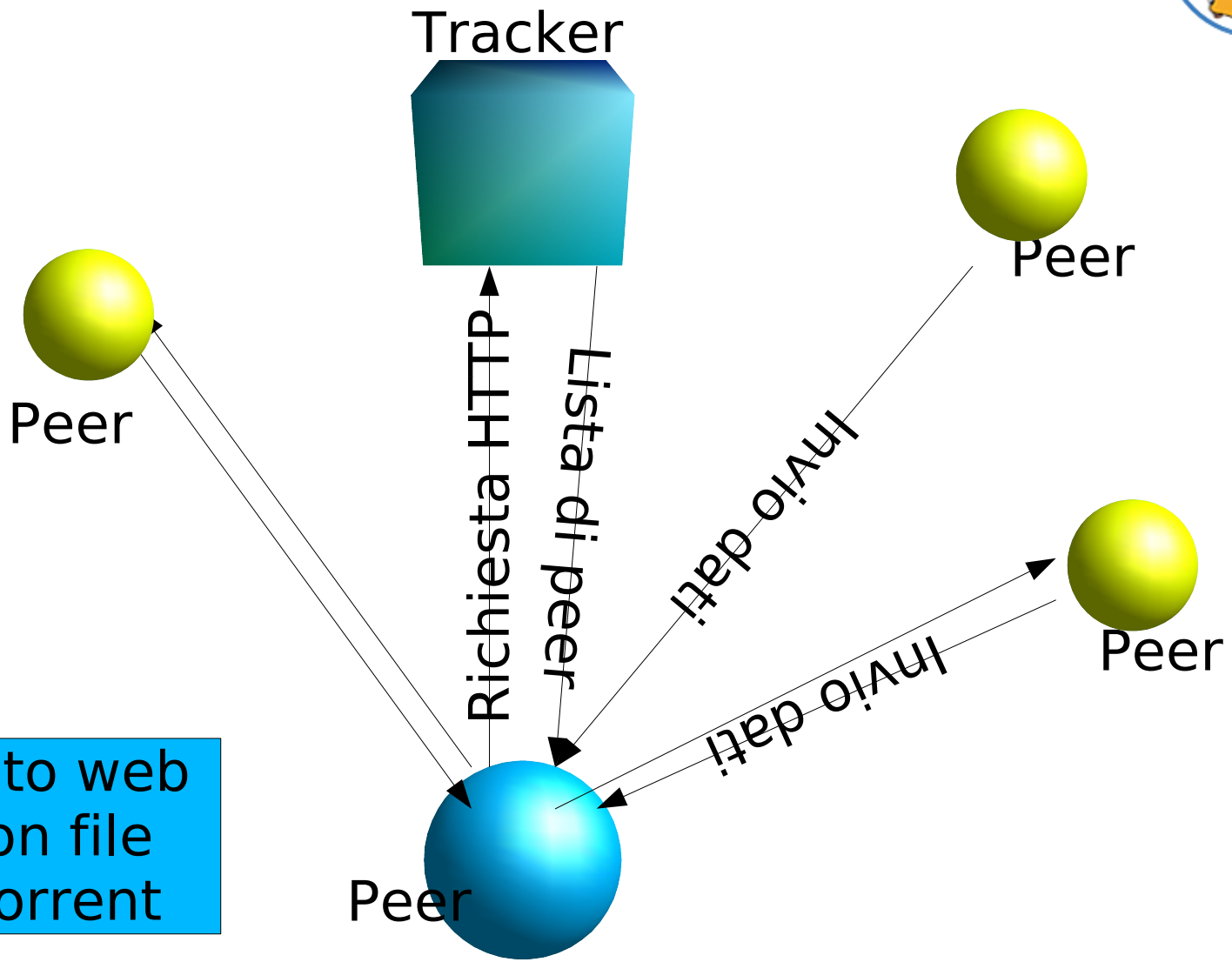
Sito web con file .torrent




Tracker
Peer
Peer
scakeand
scakehand
shakehand
Peer
Peer
Sito web con file .torrent

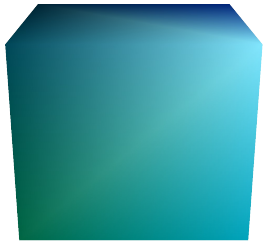

Tracker
Peer
Peer
Invio dati
Invio dati
Peer
Invio dati
Peer
Sito web con file .torrent

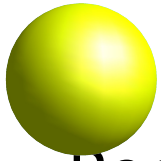

Tracker
Peer
Peer
Invio dati
Invio dati
Invio dati
Peer
Peer
Sito web con file .torrent

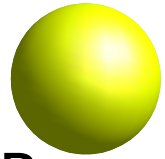

Tracker
Peer
Peer
Richiesta HTTP
Lista di peer
Invio dati
Peer
Invio dati
Peer
Sito web con file .torrent

Esempio

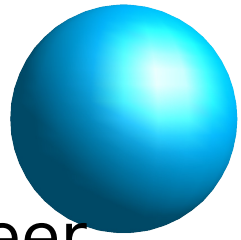
- Supponiamo che un file sia stato diviso in 10 blocchi cioè {1,2,3,...,10}
- Supponiamo di avere tre peer
- peer A possiede i blocchi 1,2,3
- peer B possiede i blocchi 4,5,6
- peer C possiede i blocchi 7,8,9,10

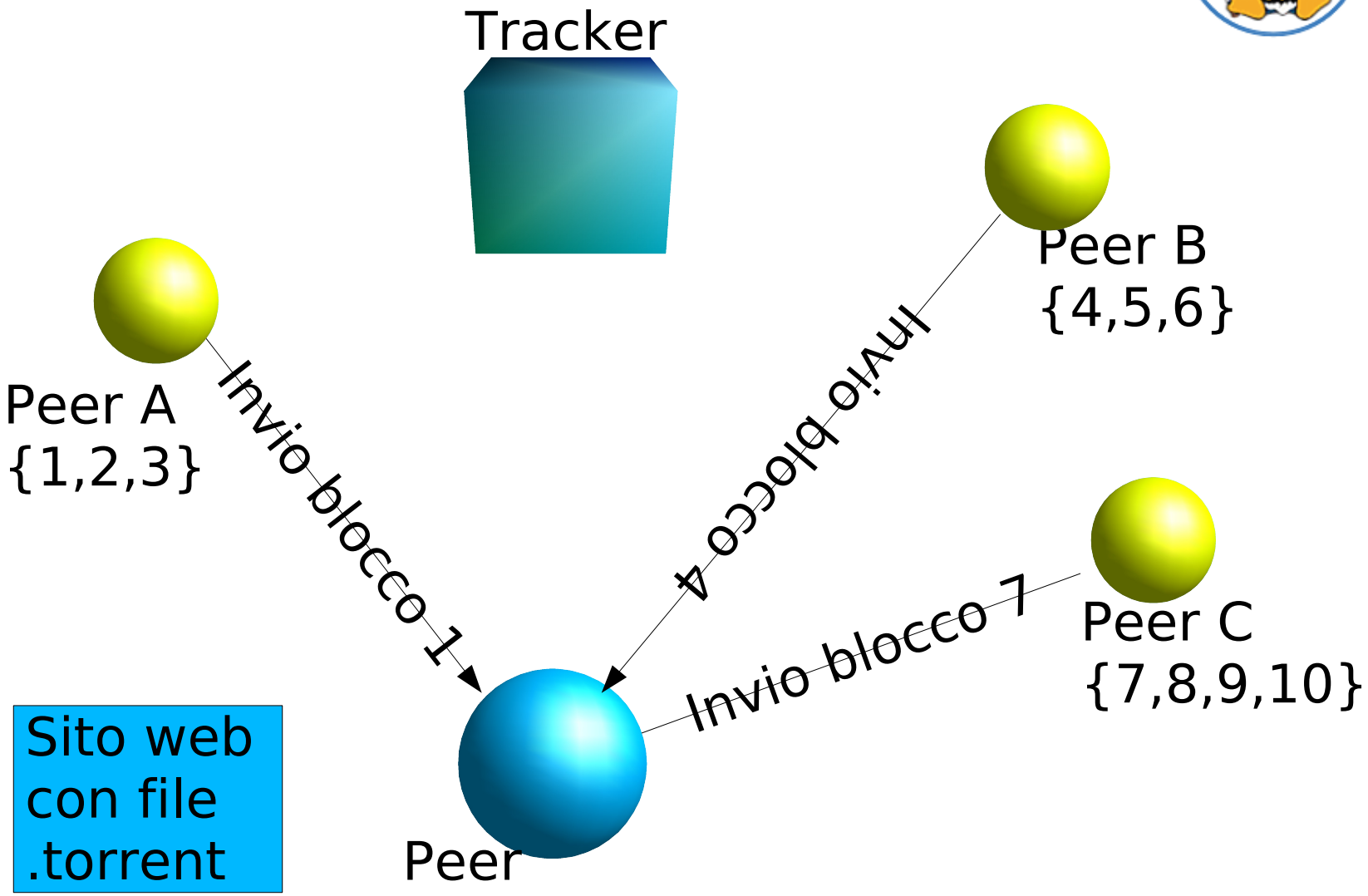




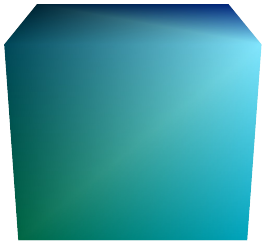


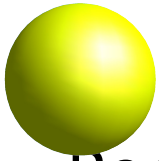
Sito web con file .torrent




Tracker
Peer A
{1,2,3}
Peer B
{4,5,6}
Peer C
{7,8,9,10}
Invio blocco 1
Invio blocco 4
Invio blocco 7
Peer
Sito web
con file
.torrent

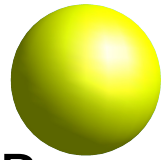
Tracker





Peer B
{4,5,6}

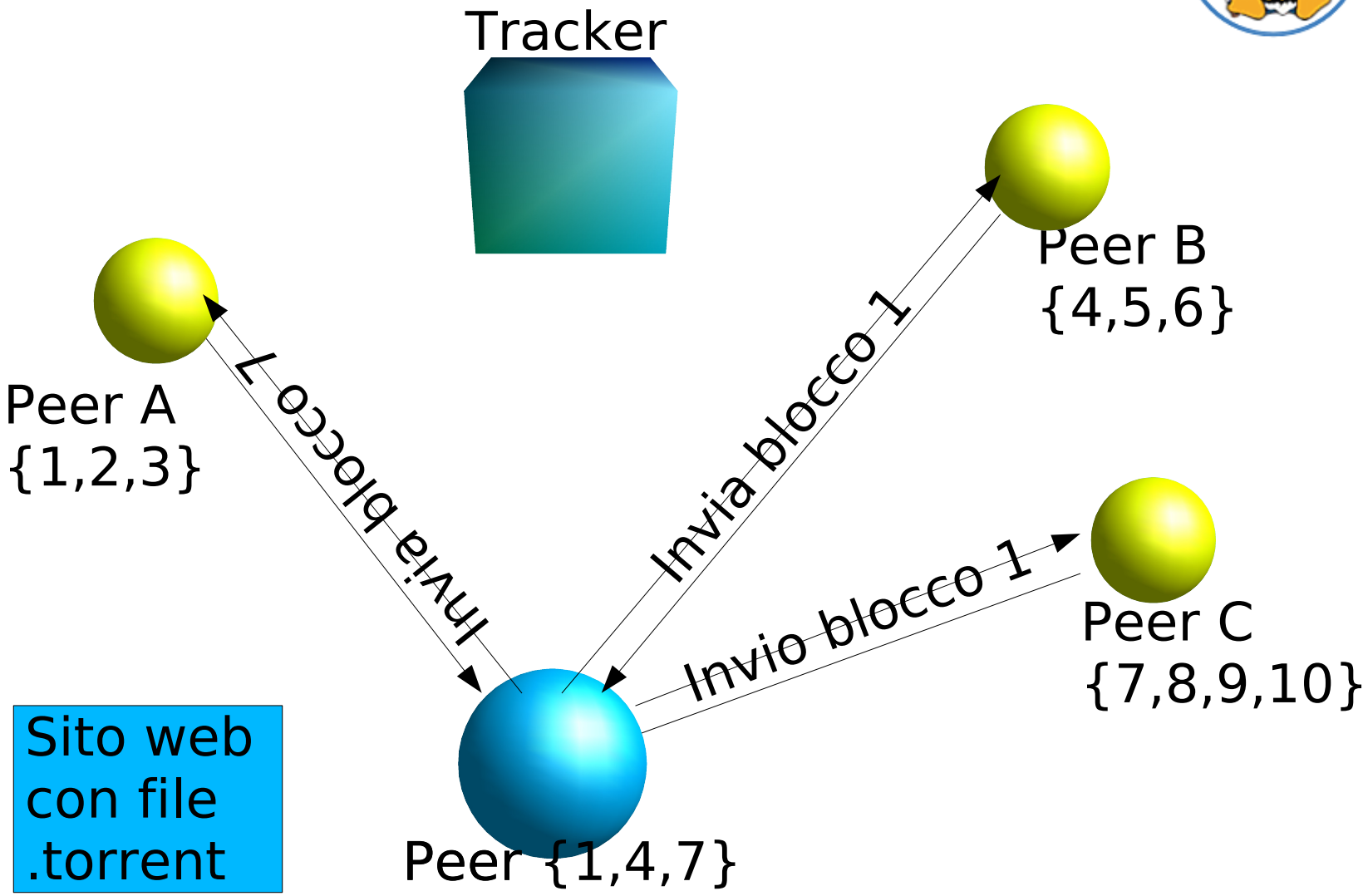
Peer A
{1,2,3}

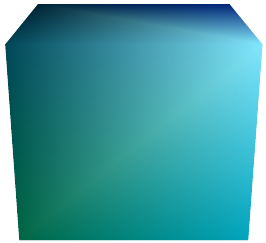


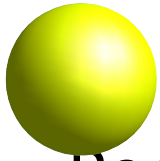
Peer C
{7,8,9,10}

Sito web
con file
.torrent

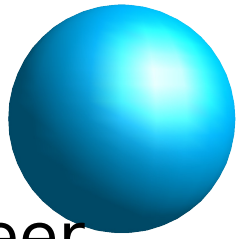
Peer {1,4,7}


Tracker
Peer A
{1,2,3}
Peer B
{4,5,6}
Peer C
{7,8,9,10}
Peer {1,4,7}
Invia blocco 7
Invia blocco 1
Invio blocco 1
Sito web
con file
.torrent





Sito web con file .torrent



- Introduzione
- Come funziona: un breve sguardo
- I componenti
- Come creare un torrent
- Come funziona: qualche dettaglio
- Selezione dei chunck
- Algoritmi di choking
- Client
- Conclusioni

Le componenti

- .torrent
  Un file che contiene le informazioni necessarie per scaricare la risorsa voluta
- Client:
  è sia colui che scarica un dato contenuto sia colui che permette di scaricarlo
- Tracker
  Coordina l'attività dei peer o client
- Qualcuno che ci fornisce il torrent:
  webserver, posta elettronica,...

.torrent

- Nome del file da scaricare
- Dimensioni del file
- L'hash di ciascun blocco in cui il file è stato diviso; in tale modo abbiamo la certezza di scaricare i blocchi di dati corretti
- La dimensione dei blocchi
- Indirizzo del tracker

Tracker

- Coordina l'attività dei peer
- Conosce l'indirizzo di ogni peer
- Di ogni peer conosce lo stato di download
- Per potere assistere meglio i peer è anche informato su quali pezzi del file siano posseduti da ciascuno
- Viene interrogato periodicamente

Client o peer

Esistono tre differenti tipi di client:

- Seeder
- Leecher
- Reseeder

Seeder

- È detto seeder quel client che possiede una intera copia del file.
- È necessario che almeno inizialmente ci sia almeno un seeder che possa *inseminare* la rete (a partire dal quale possa essere scaricato).
- In determinate circostanze, ci potrebbe non essere alcun seeder ma, malgrado ciò, sia ancora possibile recuperare tutte le parti necessarie per mettere insieme l'intero file; in tale caso si parla di copia distribuita

Leecher

- Un leecher è un client che non ha ancora una copia completa di un particolare file.
- Quando dei nuovi client cominciano a scaricare diventano leecher finché non avranno una copia completa diventando in tal modo seeder
- Normalmente è un leecher qualcuno che scarica e non fa scaricare, che prende ma non da, che, quindi, non contribuisce al buon funzionamento della rete

Leecher

- In bittorrent un leecher ha un ruolo attivo, in quanto permette l'upload
- È interesse del leeker garantire banda per l'upload in quanto la sua velocità di download dipende dalla sua velocità di upload: più si da più si riceve

Reseeder

- Sono quei client che malgrado abbiano completato il download continuano a condividere la risorsa

- Rappresentano un importante aiuto per la comunità

- Introduzione
- Come funziona: un breve sguardo
- I componenti
- Come creare un torrent
- Come funziona: qualche dettaglio
- Selezione dei chunck
- Algoritmi di choking
- Client
- Conclusioni

Come creare un .torrent

- Per creare un .torrent il client BitTorrent mette a disposizione *bymaketorrent.py*
- Su Debian esiste un wrapper a tale file: *btmakemetafile*
- *btmakemetafile* prende in ingresso un file e restituisce un file .torrent
- *btmakemetafile* crea un hash per ogni blocco
- Il valore hash vine utilizzato dal client per verificare l'integrità dei dati.

Come creare un .torrent

- *btmakemetafile nome_file tracker_announce_address [option...]*
- nome_file potrebbe essere '*presentazione_ld05.pdf*' oppure *ubuntu-5.10-install-amd64.iso*
- tracker_announce_address *è nella forma* *http://xxx.xxx.xxx:6969/announce*
- *announce* è un valore hardcoded che deve essere sempre presente
- La porta di default è la porta 80, ma viene raccomandato l'utilizzo della porta 6969

Come creare un .torrent

- Il file .torrent generato ha nome
  presentazione_ld05.pdf.torrent
  oppure
  ubuntu-5.10-install-amd64.iso.torrent
- La dimensione del file dipende dalle dimensioni del file d'origine
  - Le dimensioni del .torrent della ISO di Ubuntu sono di 25k
  - Le dimensioni di un file di 10MB è di circa 1K

Come creare un .torrent

- Sono tre, la più interessante delle quali è *piece_size_pow2 <arg>*
- Il file .torrent viene suddiviso in pezzi della dimensione di 2exp(<*arg*>)
- Di default <*arg*> è uguale a 18, quindi il file viene suddiviso in pezzi di 256MB ciascuno
- Nelle versioni precedenti di BitTorrent (sino alla 3.0.2) il valore di default era 20, quindi 1M

Che cosa contiene un .torrent

- *announce*: indirizo URL del tracker codificato come
- *creation_date*: data di creazione
- *comment*: un commento al .torrent codificato
- *created by*: il programma che ha generato il .torrent
- *info*: dizionario principale che descrive il contenuto del .torrent.

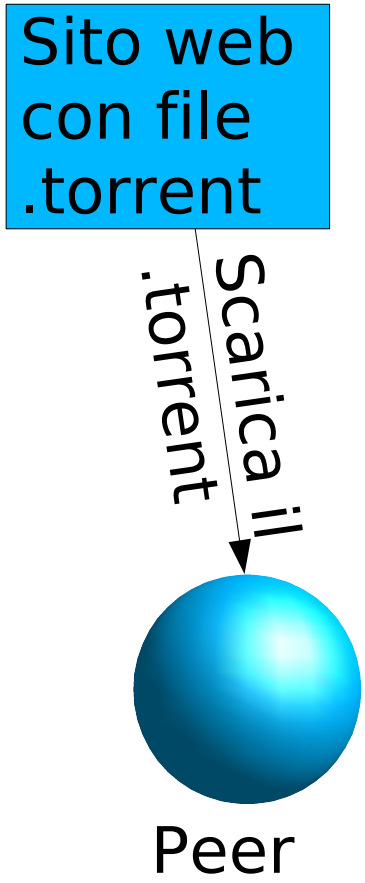
- Introduzione
- Come funziona: un breve sguardo
- I componenti
- Come creare un torrent
- Come funziona: qualche dettaglio
- Selezione dei chunck
- Algoritmi di choking
- Client
- Conclusioni

Pipelining

- È una tecnica utilizzata per aumentare la velocità di download
- BitTorrent divide i blocchi di file in sotto-pezzi, detti chunk, della grandezza tipicamente di 16k
- L'invio avviene in gruppi di chunk, solitamente in numero di 5
- Lo scopo è quello di sfruttare al massimo le connessioni (la dimensione dei chunck e la grandezza dei gruppi di chunck varia di conseguenza)

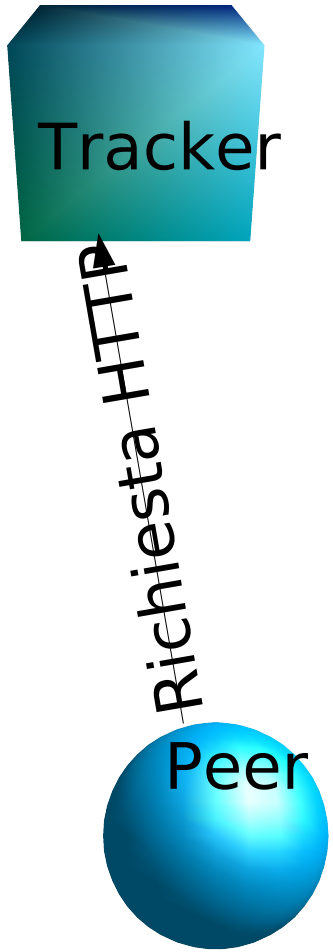
Primo passo

- Chi desidera scaricare un file deve innanzi tutto ottenere il file .torrent relativo
- successivamente “eseguire” il .torrent (cliccando sul file si avvia il client BitTorrent)



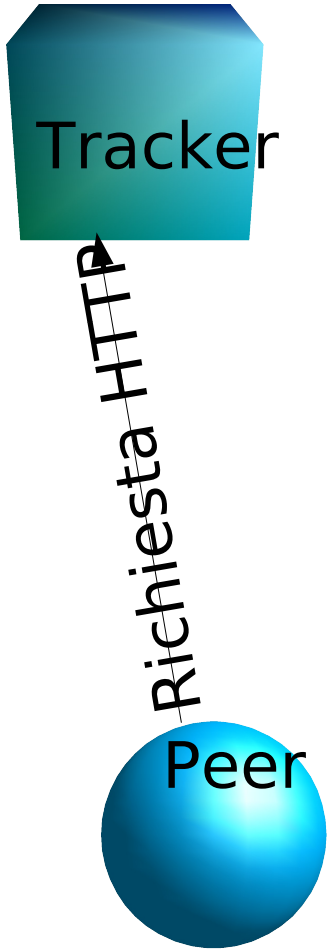
Secondo passo

- Una volta che il client BitTorrent è stato avviato
- Viene contattato il tracker indicato dal .torrent
- Lo scambio di messaggi avviene attraverso HTTP (o HTTPS) request e response



Secondo passo

- Il peer comunica ad il tracker
  - quale file vuole scaricare
  - in quale porta si attende di ricevere la lista dei peer da cui scaricare
  - Un peer_id, che lo identifica univocamente



Altri parametri

- Questi sono i parametri che devono essere inviati al tracker:
  - *info_hash*: viene utilizzato SHA1
  - *port*: numero di porta sulla quale il client è in ascolto. Le porte tipiche sono nel range 6881-6900
  - *uploaded*: bytes inviati agli altri client dall'inizio della sessione
  - *downloaded*: bytes scaricati dagli altri client dall'inizio della sessione
  - *left*: bytes rimanenti al completamento del file,
  - Se si è un seeder il valore è zero

Secondo passo

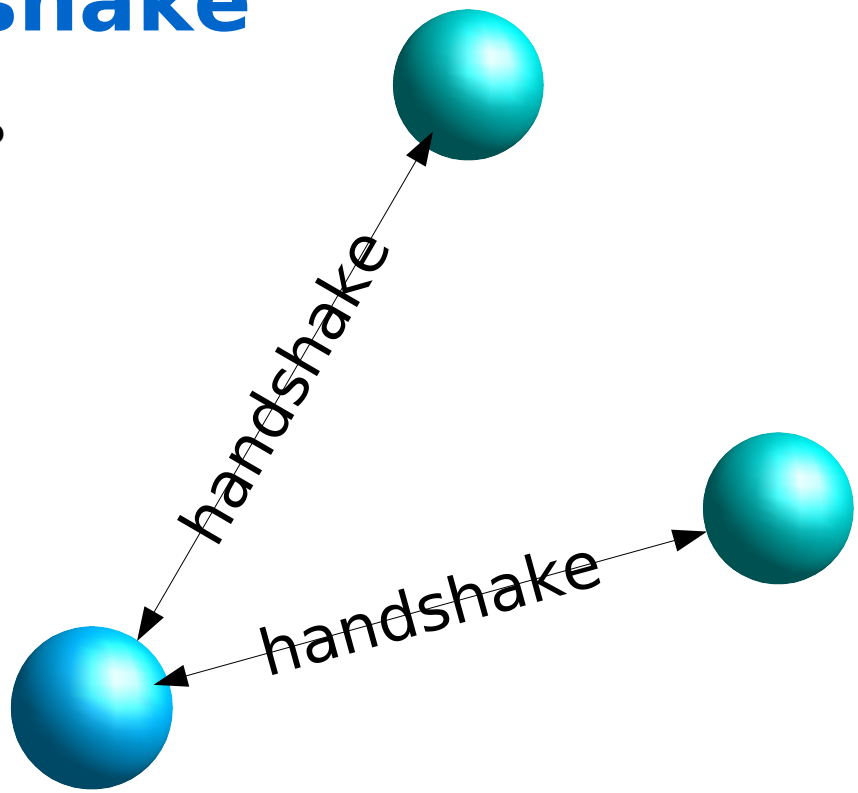
- Il tracker restituisce una lista di peer da cui è possibile scaricare
- In genere il tracker invia una lista di 50 peer

Bencode

- I messaggi vengono scambiati attraverso codifica Bencode
- Derivato dall'XML
- Quatro tipi
  - Dizionario
  - Lista
  - Intero
  - Stringa

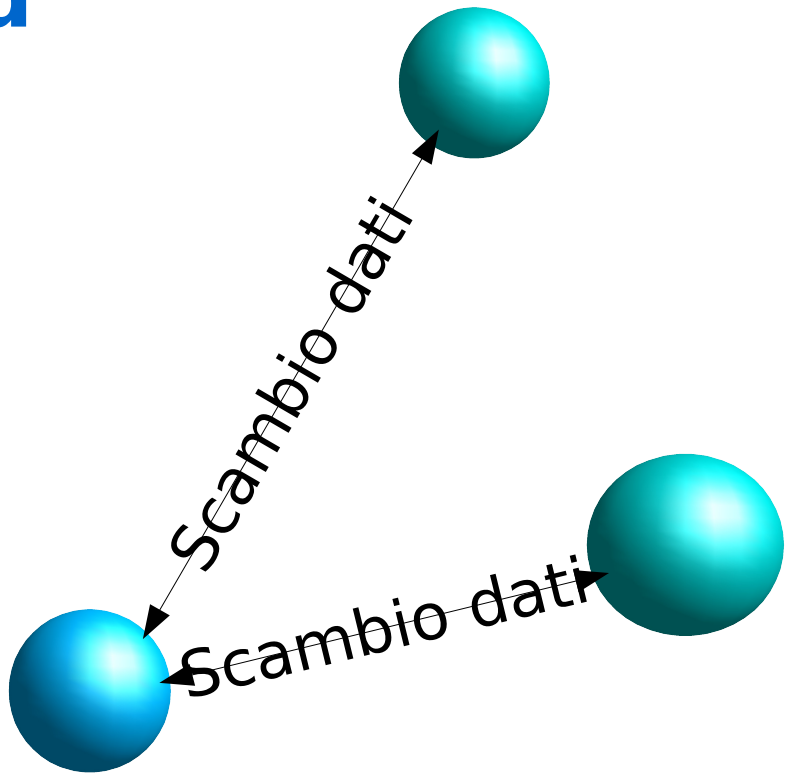
Terzo passo: Handshake

- Viene istanziata una connessione dal peer che desidera scaricare il file con i peer presenti nella lista fornita dal tracker
- I messaggi vengono scambiati attraverso TCP Socket
- Fase di handshake



Quarto passo: Bitfield

- Ogni peer informa gli altri di quali pezzetti ha a disposizione
- Viene utilizzato SAH1 per verificare l'integrità dei blocchi
- Un peer annuncia il possesso di un pezzetto di file solamente dopo averne verificato l'integrità
- Scambio di dati è reciproco



- Introduzione
- Come funziona: un breve sguardo
- I componenti
- Come creare un torrent
- Come funziona: qualche dettaglio
- Selezione dei chunck
- Algoritmi di choking
- Client
- Conclusioni

Quale file? Rarest first

- Dopo che ogni peer è in possesso delle informazioni su quali pezzetti di file sono disponibili presso gli altri peer, può ora dedurre quali blocchi siano i più rari e quindi difficili da reperire e quali siano i più diffusi e quindi più facili da reperire

- Il protocollo prevede che i primi blocchi ad essere scaricati siano quelli che con più difficoltà possono essere reperiti

Importanza di un buon algoritmo

- Immaginiamo che un blocco sia posseduto da un solo peer, se questo abbandona la rete allora sarà impossibile per tutti gli altri client completare il download; diffondendo i blocchi rari si evita questo problema

- Questo dimostra quanto sia critica la decisione su quale algoritmo di adottare

Due eccezioni

- È importante che un leek inizi il prima possibile a uplodare i blocchi che possiede in quanto la sua velocità di download dipende dalla sua velocià di upload

- Quindi quando un client non possiede alcun blocco da distribuire è preferibile che adotti un algoritmo che gli permetta rapidamente di avere un blocco completo da uplodare
  - Strict priority
  - Random first

Endgame modo

- Viene adottato quando si è prossimi a terminare il download

- Viene inviata una richiesta per i pezzi mancanti ad ogni peer

- Quando un blocco arriva deve essere inviata una richiesta in cui si esprime che quel blocco non è più di nostro interesse

- Introduzione
- Come funziona: un breve sguardo
- I componenti
- Come creare un torrent
- Come funziona: qualche dettaglio
- Selezione dei chunck
- Algoritmi di choking
- Client
- Conclusioni

Choking

- Garantisce la collaborazione tra i peer eliminando il dilemma del prigioniero

- Ogni peer cerca di massimizzare la propria velocità di download

- I peer cercano di scaricare dal maggiore numero possibile di peer, e forniscono dati ad i peer che garantiscono una alta velocità di download, gli altri vengono scartati (choke)

- choking è un rifiuto di fornire dati, ma non di riceverli

Algoritmi di Choking

- Ogni peer fornisce dati ad un certo numero di altri peer (generalmente quattro): unchocking

- Quali siano i tali peer dipende dalla velocità di download

- Ogni peer ogni 10 secondi verifica se continuare a fornire dati a quei 4 peer prescelti

Optimistic unchoking

- Un ulteriore peer rimane unchocked

- Si valutano le performance, cioè si risponde alla domanda: mi convine continuare l'upload verso gli stessi peer oppure è meglio sostituirne uno con questo nuovo peer?

- Ogni 30 secondi viene presa tale decisione

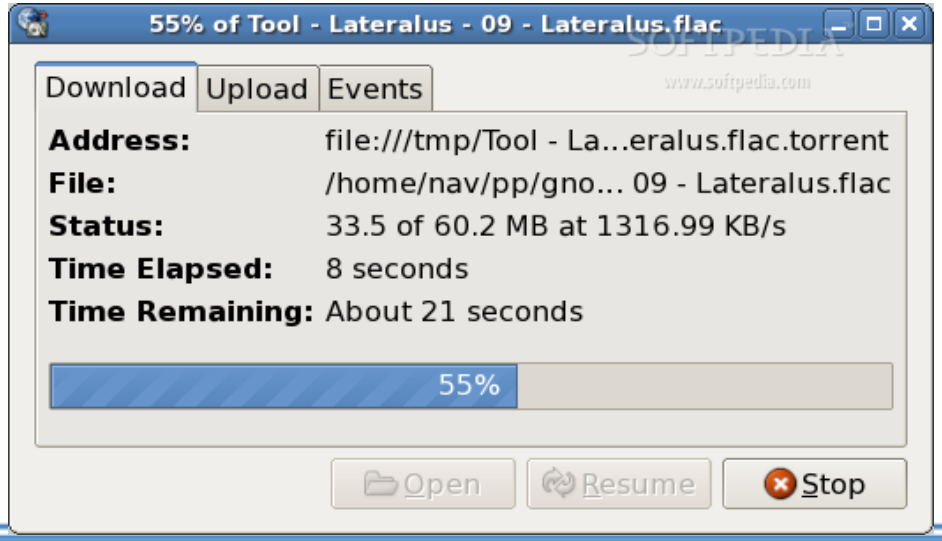
- Introduzione
- Come funziona: un breve sguardo
- I componenti
- Come creare un torrent
- Come funziona: qualche dettaglio
- Selezione dei chunck
- Algoritmi di choking
- Client
- Conclusioni

I Client opensource

- ABC
- Arctic Torrent
- Azureus
- BitTornado
- Burst
- G3Torrent
- Rufus

Gnome BitTorrent Downloader

- È un work-in-progress mime-sink per i file di BitTorrent
- NON è un front-end ma soltanto una vista di quali file .torrent si stanno eseguendo
- Scritto in python da Paul Varga



Conclusioni

- Ampiamente utilizzato
- Il caso suprnova
- In continuo progresso
- Trackerless

Grazie per l'attenzione.