



BitTorrent

INDICE

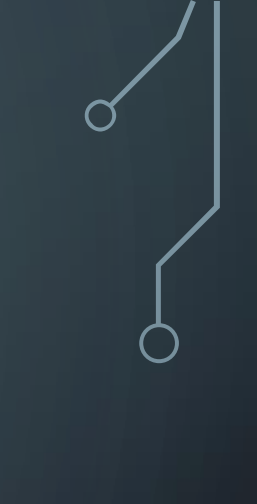
• Introduzione	5
• Architettura:	
• Schema	6
• Seed e Web Seed	7
• Peers	7
• Leecher	7
• Swarm	7
• Tracker	8
• File .torrent	8
• Client BitTorrent	8
• Magnet	8

INDICE

- Protocolli utilizzati:
 - Peer-to-Peer 9
 - Funzionamento:
 - Scambio file 10-16
 - Algoritmi utilizzati:
 - Piece download strategy:
 - Strict policy 17
 - Rarest first 17
 - Random first 18
 - Endgame Mode 18-19
 - Choking e Optimistic Unchoking 20
 - Antisnubbing 20



INDICE

- Rischi 21
 - Consigli d'uso per proteggersi 22
- 
- 
- 

INTRODUZIONE

- BitTorrent è un protocollo peer-to-peer (P2P) creato per distribuire e condividere file in Internet
- Sviluppato da Bram Cohen nel 2002 in Python
- Obiettivo: realizzare e fornire un sistema efficiente in grado di distribuire lo stesso file verso numerosi utenti che possano sia scaricarlo che condividerlo allo stesso tempo



ARCHITETTURA

- Seed: nodo che dispone di tutte le parti di un file e lo condivide con chi non ha ancora completato il download del file
- Web Seed: capacità del client di scaricare parti di torrent da una fonte HTTP
- Peer: nodo che non ha ancora tutte le parti di un file, mentre le scarica le condivide con gli altri peer
- Leecher (= sanguisuga): nodi che scaricano i file, senza condividerli e in alcuni casi possono essere bannati
- Swarm: numero complessivo di seed e peer che condividono lo stesso file

ARCHITETTURA

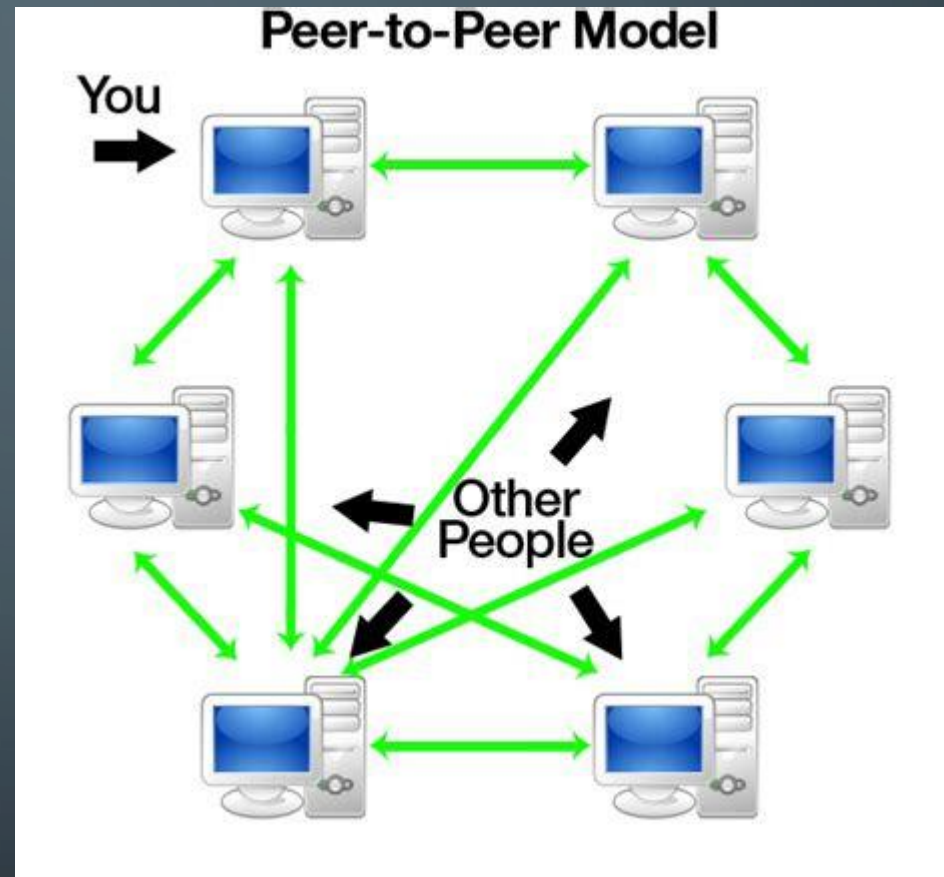
- Tracker: utilizzati per trovare i nodi che possiedono il file, alcuni client possono essere settati come tracker ed essi si occuperanno di richiedere ad altri tracker eventuali nodi
- File .torrent: file di dimensioni ridotte, contiene una lista di informazioni codificate tramite hash che descrivono i frammenti, la loro integrità, inoltre contiene tracker e web seed
- Magnet: link utili alla condivisione, si occupa di contattare i tracker per recuperare informazioni sui frammenti del file (formato link:
`magnet:?xt=urn:btih:hash(SHA1)&dn=title&tr=tracker`)
- Client BitTorrent: software in grado di utilizzare il protocollo

PROTOCOLLI UTILIZZATI

Peer-to-Peer: modello di architettura logica in cui gli host possono svolgere la funzione sia di client che di server.

Due modelli di peer-to-peer sono centralizzato e decentralizzato:

- Centralizzato: ha un server che conosce e amministra i dati all'interno dei nodi
- Decentralizzato: ogni nodo è un peer che può funzionare come client o server



FUNZIONAMENTO

Il funzionamento di BitTorrent avviene tramite i seguenti passi:

- Scambio dei dati:
 - Alcune connessioni peer-to-peer scelgono un singolo peer per lo scaricamento completo del file
 - BitTorrent scarica il file da diversi peer, dividendolo in pezzi da 256Kb o 512Kb
 - Sceglie dei pezzi tramite il 'Piece download strategy', utilizzando anche alcuni algoritmi di 'choking'
 - Obiettivo finale: replicare tutti i pezzi per arrivare al file completo

FUNZIONAMENTO: SCAMBIO DEI DATI

- Un peer instaura una connessione TCP tramite 3Way Handshake con un altro peer che è in attesa
- Questa connessione può essere troncata in caso di `peer_id` errato
- Dopo l'handshake, viene mandato il Bitfield (una struct di dati che consiste in uno o più bit allocati per motivi specifici):
 - Opzionale
 - Lunghezza variabile, indica i frammenti posseduti dal peer che manda il messaggio
 - Inizio richieste di frammenti attraverso le *request*
 - `<len=0001+X><id=5><bitfield>`

FUNZIONAMENTO: SCAMBIO DI DATI

- Request:
 - Lunghezza fissa (16 kiB), al cui interno nel cui body sono presenti le informazioni riguardanti il pezzo richiesto
 - Se uno dei peer ha le informazioni richieste, si passa all'upload del frammento attraverso un messaggio di tipo *Piece*
 - `<len=0013><id=6><index><begin><length>`
- Piece:
 - Messaggio di dimensione variabile contenente delle informazioni da scaricare per poi messi insieme formare il file completo

Parameter	Description
info_hash	20-byte SHA1 hash of the value of the info key from the Metainfo file.
peer_id	20-byte string used as a unique ID for the client, generated by the client at startup
port	The port number that the client is listening on. Ports reserved for BitTorrent are typically 6881-6889.
uploaded	The total amount uploaded so far, encoded in base ten ascii.
downloaded	The total amount downloaded so far, encoded in base ten ascii.
left	The number of bytes this client still has to download, encoded in base ten ascii.
event	If specified, must be one of started, completed, or stopped. If not specified, then this request is one performed at regular intervals.
-started	The first request to the tracker must include the event key with the started value.
-stopped	Must be sent to the tracker if the client is shutting down gracefully.
-completed	Must be sent to the tracker when the download completes. However, must not be sent if the download was already 100% complete when the client started.
ip	Optional. The true IP address of the client machine, in dotted quad format or rfc3513 defined hexed IPv6 address.
numwant	Optional. Number of peers that the client would like to receive from the tracker. This value is permitted to be zero. If omitted, typically defaults to 50 peers.

Examples					
File size	350MB			700MB	1400MB
Piece size	64kB	256kB	512kB	64kB	1MB
Number of pieces	5600	1400	700	11200	1400
Torrent size	120kB	30kB	15kB	240kB	30kB

FUNZIONAMENTO: SCAMBIO DI DATI

- Have:
 - Quando un piece viene ricevuto e scaricato, viene controllato tramite hash e la ricezione del file viene resa nota agli altri peers (connessi al client), tramite l'invio di questo messaggio *have* che ha una lunghezza fissa
 - `<len=0005><id=4><piece index>`
- Cancel:
 - Presenta la stessa struttura della request, generalmente sono inviati verso la fine del download (endgame mode), per velocizzare la fine del download manda a tutti delle request per tutti i pieces, ma allo stesso tempo invia delle *cancel* a tutti ogni volta che un pezzo già posseduto arriva
 - `<len=013><id=8><index><begin><length>`

FUNZIONAMENTO: SCAMBIO DI DATI

- Keep – alive:
 - Vengono spediti questi messaggi di 0 byte e servono a mantenere la connessione se non sono stati dati comandi per un determinato tempo (solitamente $TTL = 128$), non contiene né body né ID
 - Keep-alive: <len=0000>

FUNZIONAMENTO: SCAMBIO DI DATI

I messaggi qui presentati servono principalmente ad aggiornare lo stato della connessione:

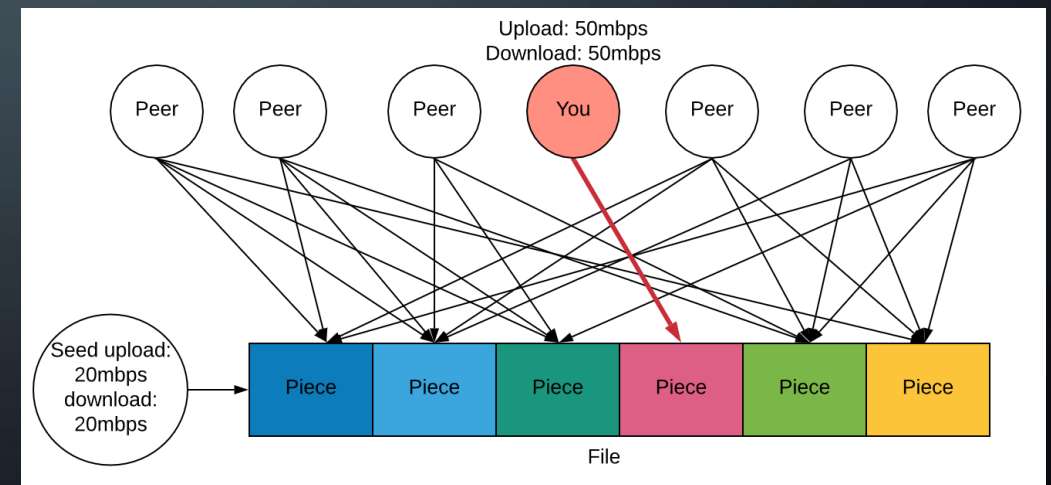
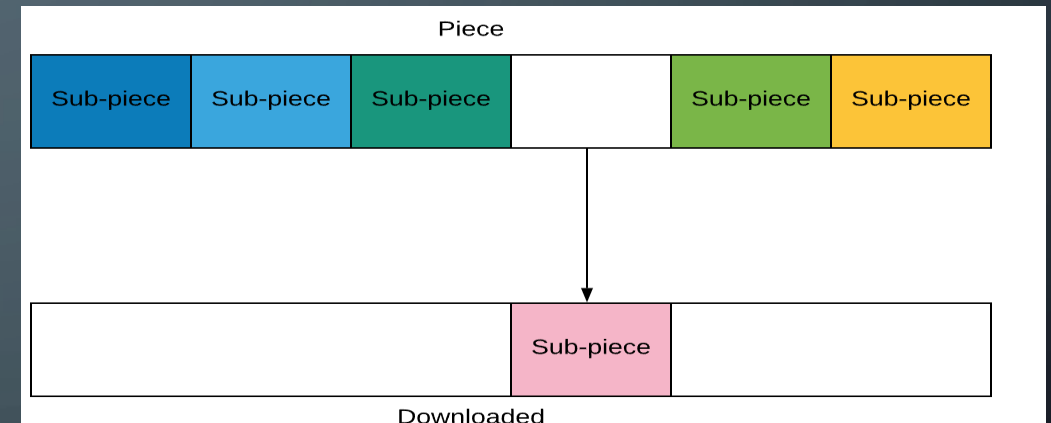
- Choke <len=0001><id=0>:
 - Messaggio con lunghezza fissata, senza body che serve a bloccare una richiesta di upload
- Unchoke <len=0001><id=1>:
 - Messaggio con lunghezza fissata, senza body che serve ad accettare una richiesta di upload
- Interested <len=0001><id=2>:
 - Se il peer remoto è interessato in qualcosa che il client ha da offrire
- Not interested <len=0001><id=3>

FUNZIONAMENTO: TABELLA TRACKER

Tracker Handshake	<length prefix>	<message ID>	<payload>
Keep-alive	0000	0	none
Choke	0001	0	none
Unchoke	0001	1	none
Interested	0001	2	none
Not-interested	0001	3	none
Have	0005	4	Piece index
Bitfield	0001+X	5	Bitfield
Request	0013	6	<index> <begin> <length>
Piece	0009+X	7	<index> <begin> <block>
Cancel	0013	8	<index> <begin> <length>
port	0003	9	<listen-port>

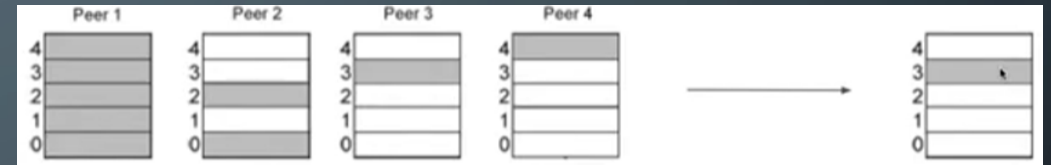
ALGORITMI: PIECE DOWNLOAD STRATEGY

- Strict policy: ogni piece viene diviso in *sub-pieces*, quando uno di questi viene richiesto, tutti gli altri sub vengono richiesti
- Rarest first: dopo il primo piece, viene scelto il piece più raro all'interno dello swarm



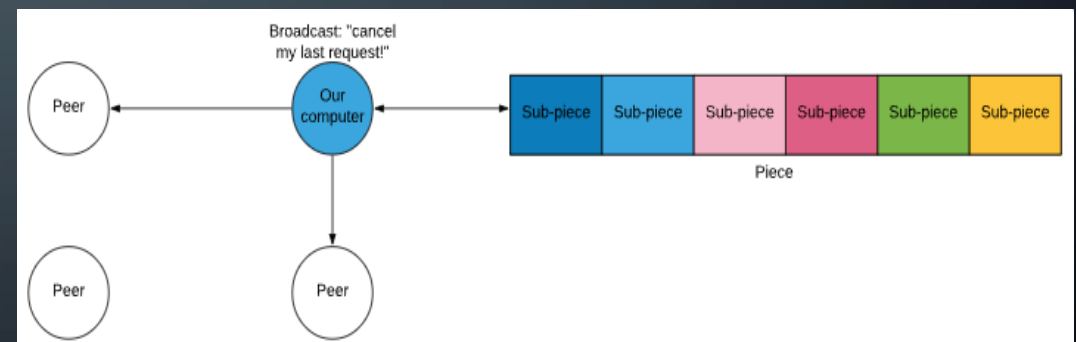
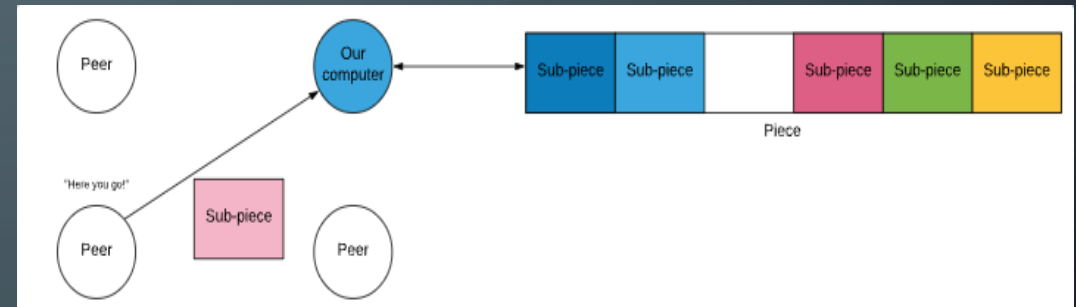
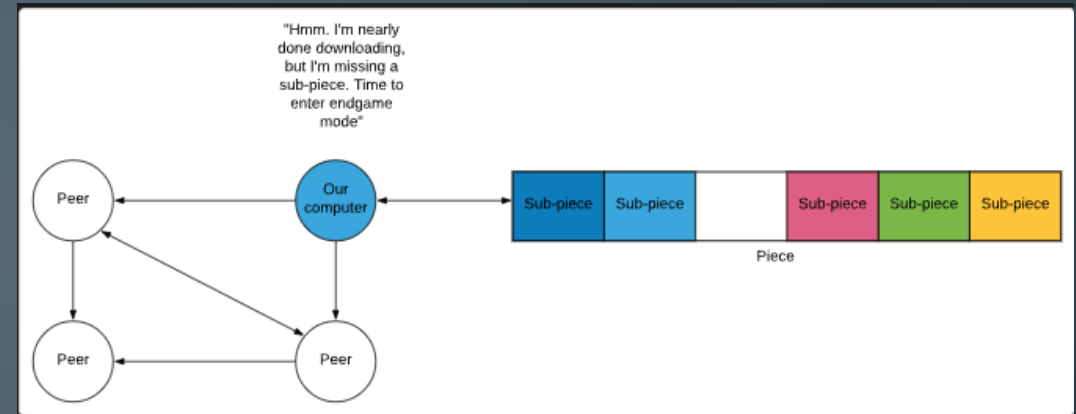
ALGORITMI: PIECE DOWNLOAD STRATEGY

- Random First: sceglie il primo file in modo randomico e poi viene applicato il Rarest First, a causa della possibilità di rallentamento
- Endgame Mode: serve a preventivare un delay dei download a causa di un download lento di un peer



ALGORITMI: ENDGAME MODE

1. Manda in broadcast una request per un *sub-piece* a tutti gli altri peers
2. Se uno dei peer ha il *sub* mancante, viene mandato al client a cui serve
3. Quando il *sub* arriva, il client manda agli altri peer un messaggio *cancel* per fare in modo che gli altri peer ignorino la request precedente



ALGORITMI: CHOKING

- Choking: pratica per cui un peer si rifiuta di inviare parti di file ad un client (solo nel caso in cui il client è un seed, è blacklistato o sta caricando a pieno regime)
- Optimistic Unchoking: peer a cui il client fa l'upload indistintamente da ogni download in corso, serve per scoprire delle connessioni migliori mai usate e per dare un servizio minimo ai nuovi peer
- Antisnubbing: se un peer non ha ricevuto nulla negli ultimi 60 secondi da un altro peer si presume sia stato "snobbato", e procederà al choking di tale peer

RISCHI

- Pericolo di download di malware, esempio TorrentLocker (ransomware)
- Pirateria di file (film, videogiochi, etc.), sito più famoso è The Pirate Bay
- Installazione di programmi dannosi da parte di alcuni client, esempio è uTorrent che installava nella macchina un software per mining di Bitcoin

PREVENZIONE

- Installazione di Antivirus
- Utilizzo di proxy o vpn per mascherare l'IP del client
- Utilizzo di block-list per gli IP malevoli
- Utilizzo di un firewall
- Scaricare i programmi voluti dai siti ufficiali/legittimi