Applicazioni P2P

Reti di Elaboratori 2017/2018
Corso di Laurea in Informatica
La Sapienza, Università di Roma

Mauro Piva

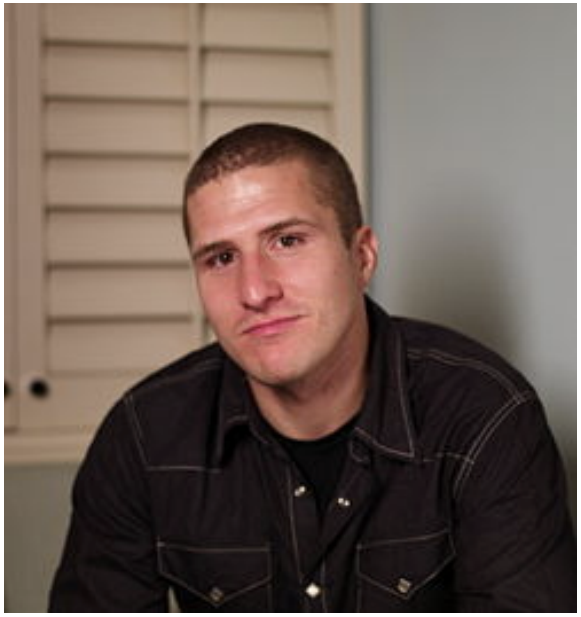
SAPIENZA
UNIVERSITÀ DI ROMA

P2P – Hall of Fame?

- Napster
- eMule
- BitTorrent
- Blockchain?

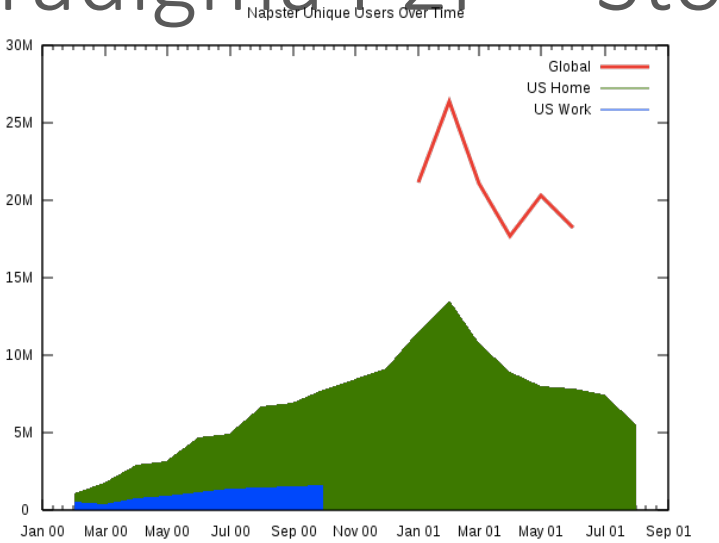
Il Paradigma P2P - Storia

- Nato negli anni 80 -> 1979 rappresenta la base per USENET



- 1999: Fanning pubblica

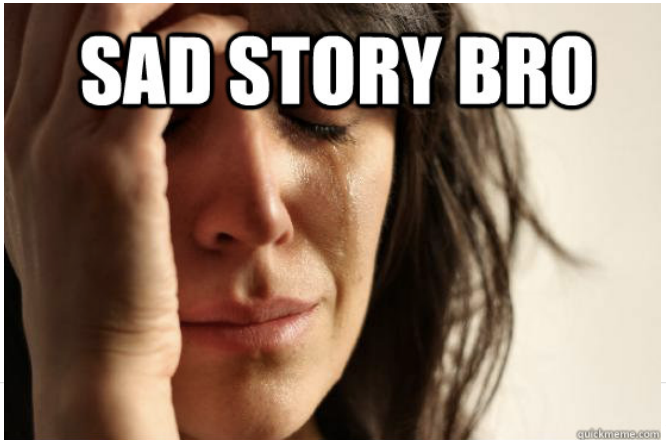
Il Paradigma P2P – Storia - napster



World Regions	Internet Users, 2000
Northern America	108,096,800
Oceania	7,619,500
Europe	103,096,093
Latin America + Caribbean	18,068,919
Asia	114,303,000
Middle East	5,284,800
Africa	4,514,400
Total World	360,983,512

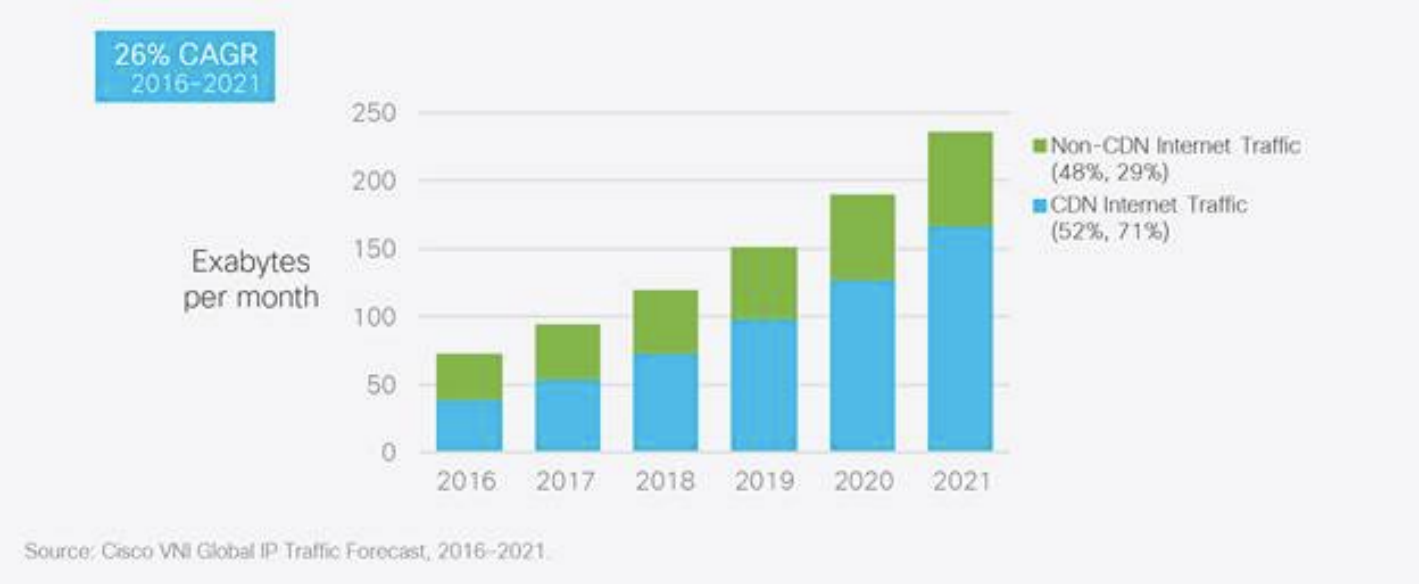
Source: Internet World Stats, July, 2005.

- 2001: Napster viene chiuso per violazione del copyright da parte di una corte americana



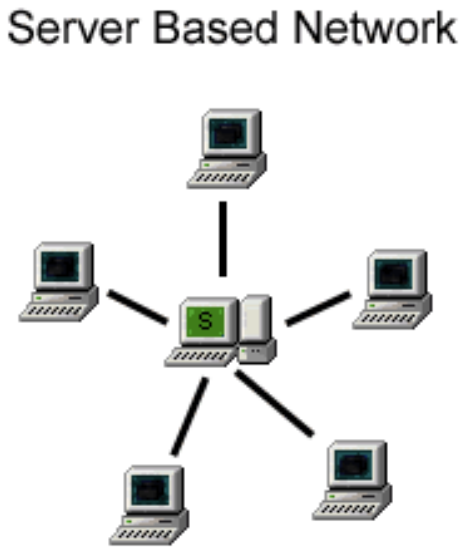
P2P – Nuovi client sviluppati

- Gnutella, Kazaa, BitTorrent, etc.
- **Febbraio 2009: 43-70% del traffico Internet viene generato da applicazioni P2P**
- Nel 2018?
- **CDN**

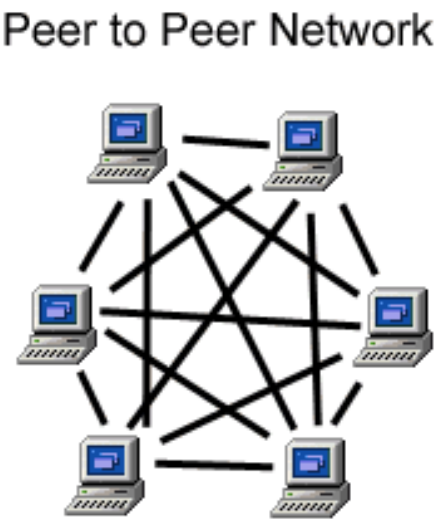


P2P vs Client/Server

- Architettura Client/Server
  - Alcuni computer sono dedicati per offrire servizi agli altri



- Architettura P2P
  - Una rete in cui tutte le workstation (i peers) hanno le stesse capacità e responsabilità



P2P

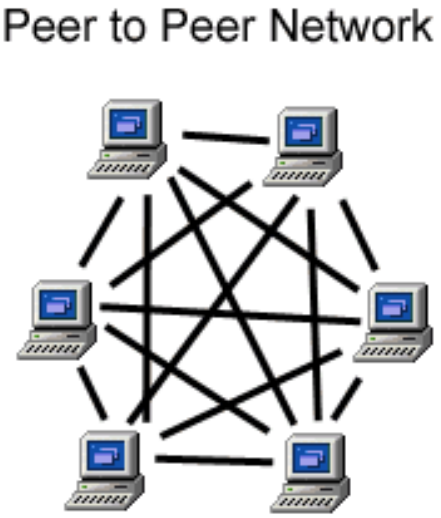
- Architettura P2P
  - Ogni peer può agire come client o come server per gli altri peer
  - Ogni peer non deve essere necessariamente sempre attivo (vedi i software P2P per il file sharing)
  - I Peer si collegano e scollegano dalla rete continuamente ed imprevedibilmente
  - Le reti P2P sono estremamente dinamiche

P2P - Obiettivi

- Alta scalabilità (scalability)
- Alta disponibilità delle risorse (resources availability)
- Essere tollerante ai fallimenti (fault-tollerant)
- Ridurre i costi
- Garantire la privacy dei peer
- Offrire un framework di rete per scenari dinamici

P2P – Sfide (Challenges)

- I peer non sono affidabili:
  - Disconnessioni impreviste
  - Poca disponibilità di banda
  - Perdita di dati locali
- I peer sono diversi (heterogeneous):
  - Diversa capacità di calcolo
  - Diversa capacità di storage locale
- Come individuiamo gli altri peer? (Resource Discovery)
- Sicurezza ed integrità delle risorse



P2P – Classificazione

- Reti non strutturate (Unstructured Networks)
  - e.g. GNUTella, Kazaa
  - Non impongono nessuna topologia per la rete dei peer (no topology on the overlay network)
  - I peer si collegano casualmente tra loro
  - La ricerca viene effettuata inondando (flooding) la rete con queries
- Reti strutturate
  - e.g. Kad, alcune modalità di BitTorrent
  - Tipicamente mediante una DHT (Distributed Hash Table) è possibile effettuare ricerche efficienti su quali peers hanno alcuni contenuti.

P2P – Classificazione – Reti Ibride

- Basate su P2P ma che includono una parte client/server
  - Semplificano il problema dell'accesso alla rete da parte di un peer (bootstrap problem)
  - Migliorano la resource discovery (evitano il flooding)
- Napster: un insieme di server possiedono un indice delle risorse (dei file condivisi da ogni peer)
  - PRO: Resource discovery semplificata
  - CONTRO: Unico punto di fallimento
  - CONTRO: Costo per l'infrastruttura server centrale e performance bottle-neck

P2P – Classificazione – Reti gerarchiche

- Prevedono l'esistenza di alcuni peers speciali chiamati "super peer" che offrono funzionalità aggiuntive
  - Scelti tra i nodi più potenti, semplificano la resource discovery senza server centrali
  - Ogni peer è connesso ad un super peer. Se il super peer non possiede le informazioni richieste dal peer le richiede agli altri super peer.

P2P – Classificazione – Reti gerarchiche

- Skype: i super peer vengono definiti “supernodes”
  - Selezionati tra i nodi più potenti, con maggiore uptime e con connettività più aperta (no firewall/NAT)
  - Mantengono informazioni su tutti gli utenti connessi a Skype e sulle comunicazioni inter nodo.
  - 2010 -> un bug (nel meccanismo di elezione dei supernodes?) crasha l’intera rete skype. Prima che l’aggiornamento raggiungesse un numero sufficiente di nodi sono state necessarie diverse ore, provocando di fatto un lungo periodo di down di Skype.
  - 2011 -> Skype sposta i supernodes su alcuni server dedicati, in grado di offrire maggiore affidabilità e controllo.

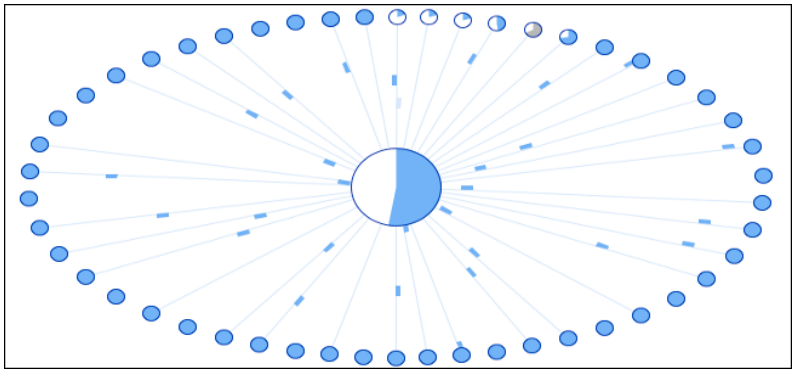
P2P – Applicazioni - BitTorrent

- Protocollo basato su P2P utilizzato per la condivisione di file su Internet
- Rilasciato (in Python) nel 2001 da Bram Cohen
- Gennaio 2012: 150 milioni di utenti attivi
- Gennaio 2018: ???
- Usato per distribuire grandi quantitativi di dati su Internet: files, disto linux, data sets, etc.

P2P – Applicazioni -

BitTorrent™

- Per ogni file da condividere viene creato un file .torrent
- I peers contemporaneamente
  - Scaricano le parti di file che vogliono ottenere
  - Fanno upload delle parti di file che possiedono o che stanno scaricando
- L'insieme dei peer attivi in un torrent viene chiamato "swarm"



P2P – Applicazioni -

- Il file .torrent contiene:
  - announce:
    - Trackers
  - info:
    - File name, File Hash, Metadata
    - Hash delle parti dei file da condividere (usati per verifica integrità)

P2P – Applicazioni - BitTorrent

- Tipi di peers:
- Per ogni torrent, l'insieme di peers viene diviso in due parti:
  - Seeders
    - Tutti i peers che hanno una copia completa del file e che continuano a condividere il file con altri peers
  - Leechers
    - Tutti i peers che non hanno una copia completa del file

P2P – Applicazioni - BitTorrent

- Come trovare i peers per un file? The Torrent way

- Ricerca sul web il file .torrent, oppure lo riceve
- Individua, nella sezione announce, gli ip dei tracker
- Interroga i tracker (scrape)
- Riceve gli IP di 50 peers che possiedono il file

P2P – Applicazioni - BitTorrent™

- I Trackers
  - Non possiedono alcuna parte del file!
  - Mantengono informazioni sui peers attualmente attivi
  - I peers aggiornano il loro stato presso i trackers ogni 30 minuti, oppure quando si uniscono o lasciano lo swarm.
  - I nuovi client ricevono da ogni tracker gli IP di 50 peers scelti casualmente

P2P – Applicazioni -

- La connessione ai peers:
  - Ricevuta la lista degli IP dei peers, il client tenta di connettersi mediante TCP a tutti i peers, generando il ”Peer Set”
  - Il Peer Set cambia nel tempo, e se i peers presenti diventano <20, il client ricontatta i trackers

P2P – Applicazioni -

- Come trovare i peers per un file? The Magnet Link Way
  - Ricerca sul web il link magnet
  - magnet:? (PROTOCOLLO)
    xt=urn:btih:9788cab754a712618671d5f526cd234c83ad94f5 (HASH)
    &dn=Young.Sheldon.S01E15.HDTV.x264-SVA (FILE NAME)
    &tr=udp%3A%2F%2Ftracker.leechers-paradise.org%3A6969 (TRACKER)
  - Se non c'è nessun tracker?
    - Distributed Hash Table
    - Bootstrap DHT?
    - Ibrido: e.g. router.bittorrent.com

P2P – Applicazioni -

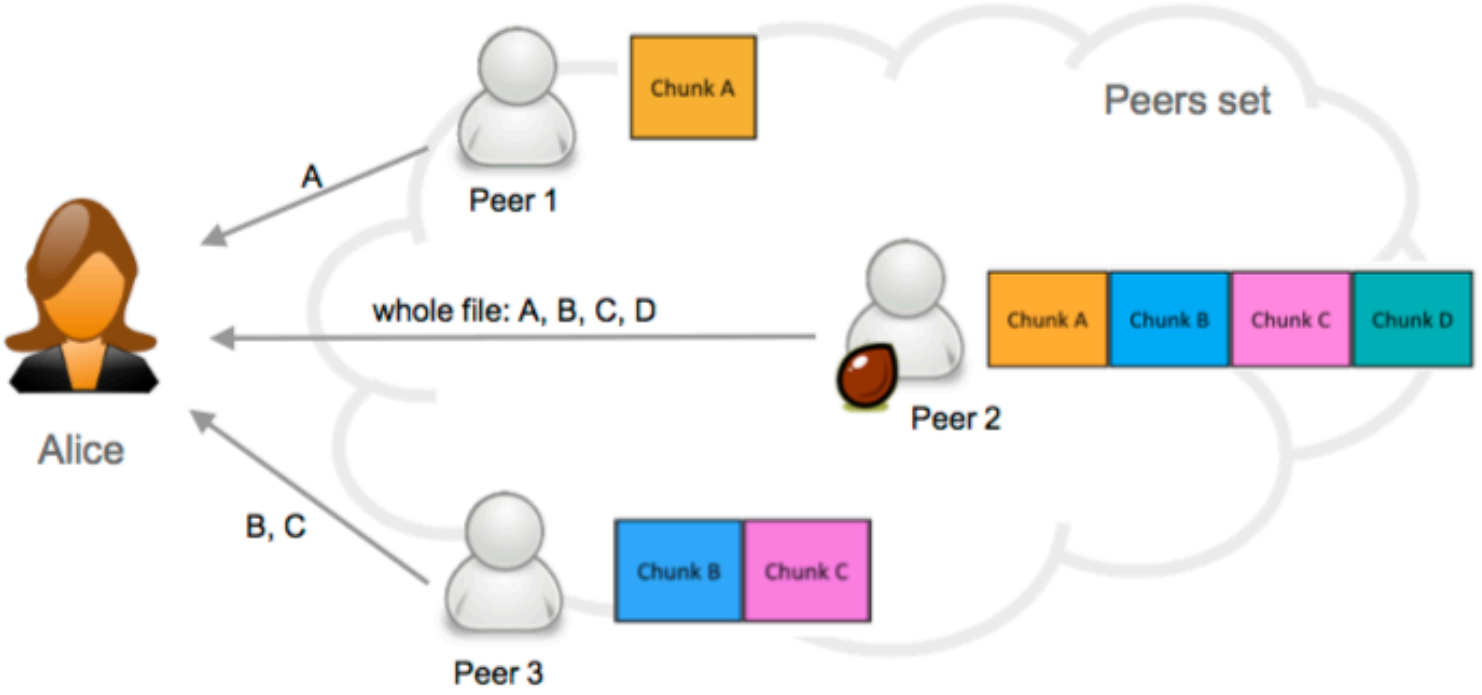
- La connessione ai peers:
  - Ricevuta la lista degli IP dei peers, il client tenta di connettersi mediante TCP a tutti i peers, generando il ”Peer Set”
  - Il Peer Set cambia nel tempo, e se i peer presenti diventano <20, il client ricontatta i trackers

P2P – Applicazioni -

- File chunks
  - Il protocollo BitTorrent prevede che i file vengano divisi in parti (**chunks**) di dimensione variabile tra 64KB ed 1MB (tipicamente 256KB)
  - Quando un peer (Alice) inizia a scaricare un file, non possiede **nessun chunk di quel file.**
  - Ogni Seeder possiede tutti I chunck del file, mentre i leechers ne possiedono solo un sottoinsieme.
  - Il client (Alice) periodicamente richiede ai peers nel ”peer set” la lista dei chunks che possiedono

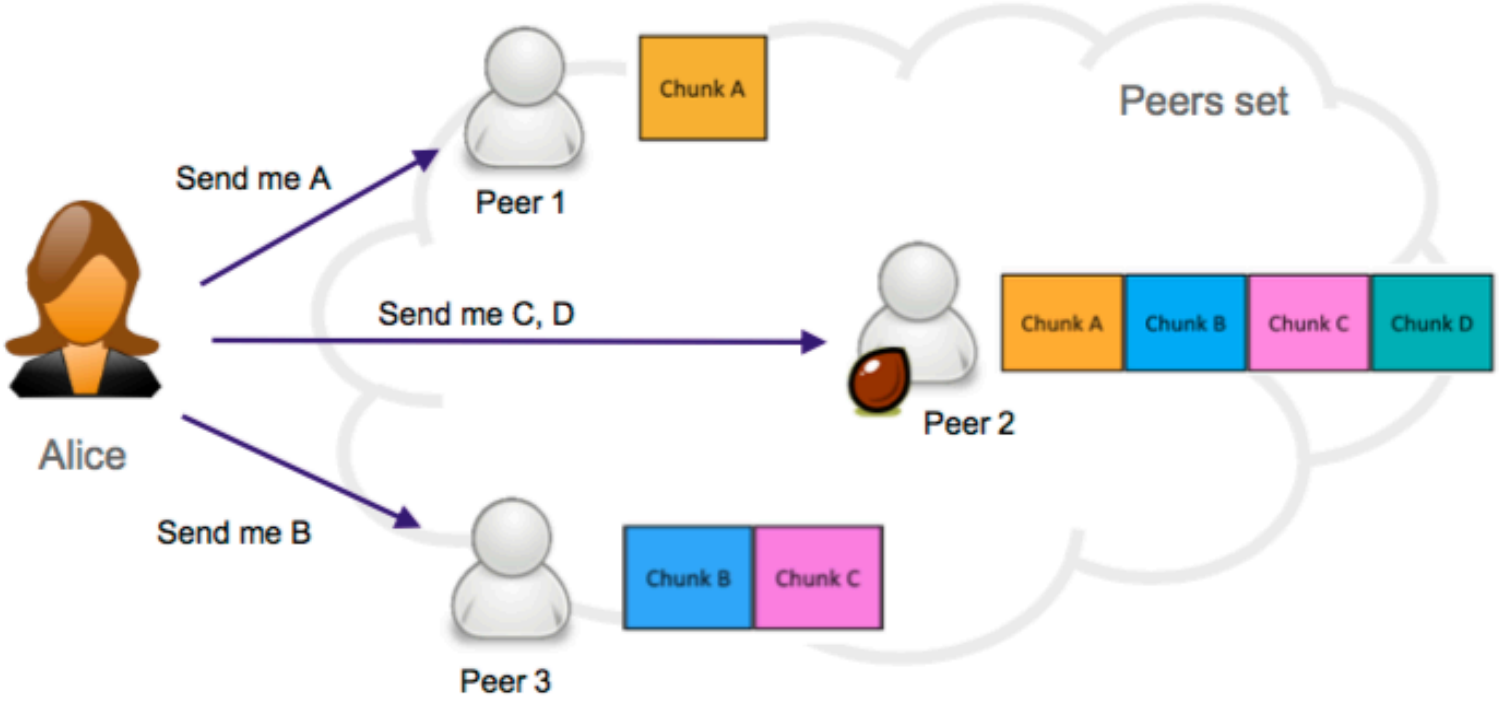
P2P – Applicazioni -

- Richiesta dei chunks



P2P – Applicazioni -

- Download simultaneo dei chunks



P2P – Applicazioni - BitTorrent

- Alice scarica simultaneamente diversi chuncks da diversi peers, tenendo traccia di quanto ha scaricato da ogni peer
- In che ordine vengono scaricati i chunks?
- **Local rarest first:** in base alla lista dei chunks ricevuta dai peers, Alice determina quali chunks sono meno diffusi, avviando il download del chunck più raro.
- I chuncks meno rari vengono scaricati successivamente
- Replicando immediatamente I chunks più rari la rete cerca di mitigare il rischio che non vi sia nessun peer connesso in possesso di quei chunks.
- Eccezione: in caso di download appena iniziato, **Random First**

P2P – Applicazioni - BitTorrent

- Appena Alice completa il download del primo chunck, è in gradi di eseguirne l'upload verso altri peers.
- Alice, avendo una banda limitata, può servire un numero limitato di altri peers.
- Come defisce a quali peer effettuare l'upload?
- **Tit for tat:** Scambia banda in upload con banda in download

P2P – Applicazioni - BitTorrent™

- **Tit for tat**:
  - Alice continuamente calcola il suo download rate rispetto a gli altri peers.
  - Alice effettua l'upload dei chunks verso i 4 peers da cui lei sta scaricano più velocemente
  - Ogni 10 secondi Alice ricalcola ed aggiorna i 4 top peers
  - Ogni 30 secondi Alice sceglie un peer casuale a cui inviare I propri chunks.

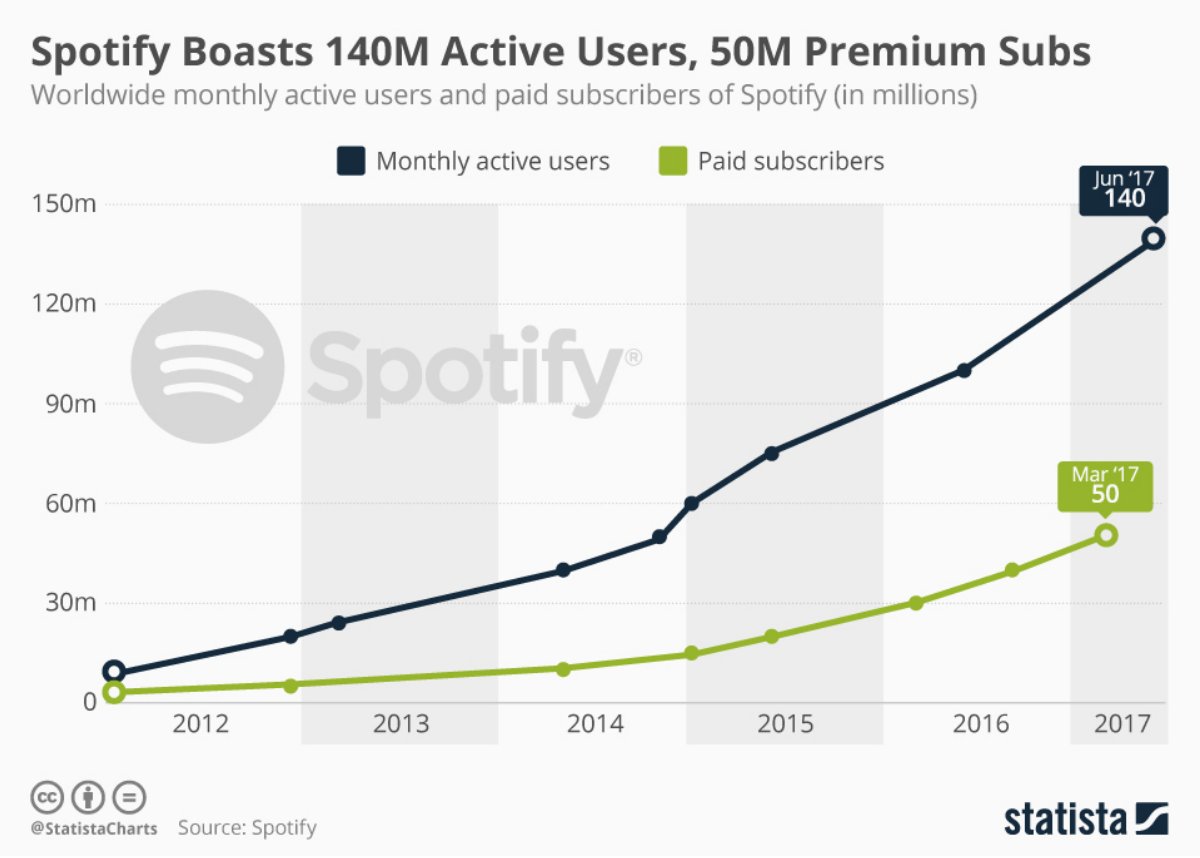
P2P – Applicazioni - BitTorrent™

- **Choking & Unchoking**
  - I 5 peers (4 top peers ed 1 casuale) vengono definiti “Unchoked”.
  - Tutti gli altri peer nello swarm sono invece “choked”, in quanto non ricevono alcun chunk da Alice
  - L’unchoking casuale viene eseguito ogni 30 secondi per:
    - Assicurare agli ultimi peers la possibilità di partecipare allo swarm
    - Potenzialmente scoprire partners più veloci

P2P – Applicazioni - BitTorrent™

- **Vantaggi**
  - Sfrutta i chuncks, consentendo download parziali
  - Incoraggia tutti i peers a condividere il più possibile
  - E' molto rapido per file di grandi dimensioni molto condivisi
- **Svantaggi**
  - Alta latenza e molto overhead per file di piccole dimensioni
  - Poco pratico per file non molto condivisi
  - Non supporta lo streaming di files
  - Ha il problema dei leech
  - Non è P2P puro: I trackers rappresentano un single point of failure.

P2P – Spotify



- Nasce nel 2008, basato su P2P
- Nel 2014 si sposta verso AWS Cloud Front (CDN)
- Nel 2016 inizia a trasferirsi verso Google Cloud

P2P – Spotify – 2008<->2014

- Sfrutta un sistema di distribuzione **ibrido** basato su:
  - Sistema client/server
  - Rete P2P dei clienti del servizio
- **Vantaggio:** solo l'**8% del traffico** proviene dai server centrali!
- Il resto del traffico viene condiviso dai peers
  (solamente i sistemi pc: smartphone e altri device sono esclusi da p2p)
- **Svantaggi?**
  - **Latenza dovuta all'individuazione dei peers**
  - **Design infrastruttura complesso**

P2P – Spotify – 2008<->2014

- Una rete P2P non strutturata:
  - La rete viene costruita e mantenuta da **trackers**, come BitTorrent, ma in questo caso si trovano sui server privati di Spotify
  - **Nessun supernodo:** tutti i nodi che partecipano alla rete P2P hanno le stesse funzioni
  - **Non utilizza Distributed Hash Table** per individuare altri peers/contenuti
  - I messaggi di discovery vengono inviati fino a due hop di distanza.
- **Vantaggi:**
  - Mantiene il protocollo semplice
  - Riduce le latenze
- **Possibile perché Spotify possiede una CDN centralizzata come backend, a differenza delle altre reti P2P (e.g. BitTorrent)**

P2P – Spotify – 2008<->2014

- Sfrutta una **Cache** locale:
  - Tutti i Client Spotify salvano le tracce già scaricate in una cache nel disco locale. Di default, la cache è pari al 10% del disco locale, ma sempre tra[50MB,10GB]
- Oltre il 50% dei client ha una cache di 5GB
  - **Vantaggio:** grande probabilità che un client ottenga una traccia da un altro client in P2P, diminuendo il carico sui server Spotify
  - **Vantaggio:** poche possibilità che un client riscarichi un brano già scaricato
  - **Svantaggio: impatto sui dischi locali degli utenti**
    - Utilizza una policy di cache-eviction LRU, Least Recently Uses, che in questo caso diventa Least Recently Played

P2P – Spotify – 2008<->2014

- Condivisione delle tracce
  - Un client può condividere solamente le tracce che possiede interamente, a differenza di BitTorrent
  - Nessun chunck, con una notevole semplificazione del protocollo
  - Riduzione del numero di client con dati disponibili, ma essendo le tracce relativamente piccole l'impatto è minimo

P2P – Spotify – 2008<->2014

- Come individuare gli altri peers?

- Chiedendo ai trackers
- Chiedendo agli altri peers connessi

P2P – Spotify – 2008<->2014

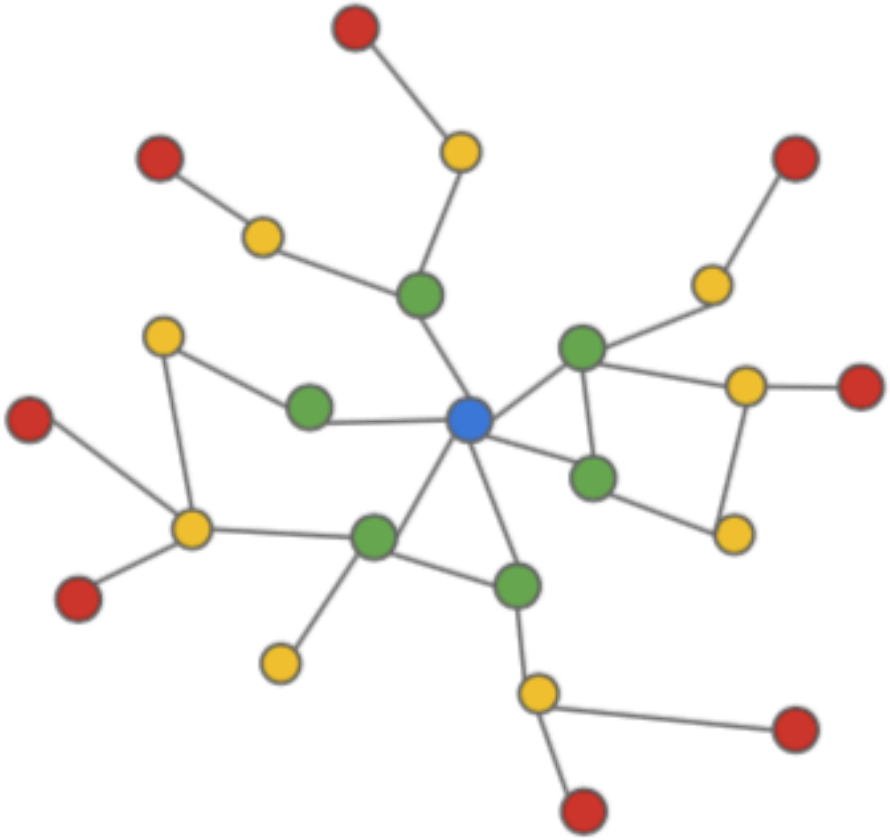
- Spotify Trackers
  - Per bilanciare il carico tra i diversi trackers, ogni peer ne seleziona uno randomicamente a cui connettersi
  - Ogni tracker è responsabile di una rete di client P2P separata ed indipendente
    - Vantaggio: non richiedere la gestione della consistenza tra i diversi trackers
    - Vantaggio: L'architettura scala facilmente: se si aggiungono nuovi cliente , è sufficiente aggiungere un tracker per creare una nuova rete P2P separata
  - In questo caso considereremo l'esistenza di un solo tracker

P2P – Spotify – 2008<->2014

- Spotify Trackers
  - I tracker server di Spotify agiscono similmente a quelli di BitTorrente, ma:
  - Non mantengono traccia di tutti peers che possiedono una traccia
  - Mantengono una lista dei 20 clients che hanno eseguito la traccia più recentemente.
  - I clients non informano i trackers server riguardo il contenuto delle loro caches.
- Vantaggi
  - Meno risorse necessarie dal lato server
  - Implementazione semplificata dei trackers
- Svantaggi:
  - solo una parte dei peers possono essere individuati attraverso il tracker
  - I Client possono richiedere agli altri peers

P2P – Spotify – 2008<->2014

- Peers



client

neighborhood

two-hops-distant peers

other peers

P2P – Spotify – 2008<->2014

- Ogni Client è connesso ad un insieme di **neighbors** (altri Client) nella rete P2P
  - I Neighbors sono i peers da cui il client ha già scaricato o a già inivato almeno un brano
- Quando un client deve scaricare una nuova traccia, cerca tra l suoi neighbors chi la possiede nella propria cache.
- I peer possono inoltrare le richieste ricevute verso i propri peer, fino ad un massimo di due hop dall'origine
- Ogni query ha un ID univoco, ciò permette di ignorare query duplicate

P2P – Spotify – 2008<->2014

- Ogni client invia dati ad un massimo di 4 client in contemporanea
  - Limita il traffico generato e la banda occupata da Spotify
- Al termine di un upload/download connessione TCP **NON viene chiusa**
  - **Riduce il tempo di connessione verso I peer quando una nuova traccia deve essere scaricata**
  - **Costoso per i peers, che devono mantenere le connessioni attive**
- Per limitare l'overhead generato, I client hanno un **soft limit** ed un **hard limit**
  - Quando viene raggiungo il soft limit (di connessioni aperte verso gli altri peers), il client smette di aprire nuove connessioni, ma continua ad accettarne
  - Quando viene raggiunto l'hard limit, non vengono ne aperte ne accettate nuove connessioni

P2P – Spotify – 2008<->2014

- Come chiudere connessioni quando vengono raggiunti I limiti?
  - Per ogni peer connesso, viene calcolato un valore **utility**, basato su
    - La quantità di dati trasmessi/ricevuti nell'ultimo periodo
    - Il numero di altri peers che il peer ha permesso di individuare
- I peers vengono ordinati per questo valore, e gli ultimi vengono disconnessi

P2P – Spotify – 2008<->2014

- Cosa avviene quando Spotify deve eseguire una traccia?
- Obiettivo principale: **bassa latenza di playback**
  - La latenza di playback è il tempo che l'utente deve attendere prima che la traccia venga eseguita ininterrottamente (come il tempo di buffering)
- Circa il 61% delle tracce viene eseguito in un ordine prevedibile (e.g. stesso album)
  - E' possibile ridurre la latenza prevedendo quali canzoni verranno richieste
- Il 39% delle tracce viene eseguito in ordine randomico
  - Difficile prevedere quali tracce verranno eseguite

P2P – Spotify – 2008<->2014

- Esecuzione delle tracce randomiche: solo P2P?
- Impossibile garantire una bassa playback latency!
  - La ricerca tra I peers richiede tempo (principalmente a causa dei molti messaggi che è necessario scambiare)
  - Alcuni peer hanno scarsa capacità di banda in upload
  - Alcuni brani potrebbero essere assenti tra i peer a cui siamo connessi
  - Una connessione TCP richiede del tempo prima di raggiungere la massima velocità (Perché?)
  - Le connessioni P2P sono inaffidabili (e.g. l'utente da cui stiamo scaricando il brano potrebbe disconnettersi)

P2P – Spotify – 2008<->2014

- Esecuzione delle tracce random: idee?
- Usare la Content Delivery Network di Spotify!
  - Rapida ed affidabile
  - Più traffico sui server Spotify -> Maggiori costi!

- Altre idee?
- Usare la CDN di Spotify solo per i primi 15 secondi della traccia
  - Concede del tempo (15 secondi) ai client per individuare il peer da cui scaricare il brano
  - La CDN di Spotify viene utilizzata solo in condizioni critiche e per parti limitate di brani

P2P – Spotify – 2008<->2014

- Esecuzione delle tracce prevedibili
  - Nel 61% dei casi, l'utente ascolta o una playlist o un album, consentendo ampi margini temporali per il prefetch della canzone successiva
- Nel 39% dei casi l'utente cambia canzone!
  - Spreco di banda e risorse
- Soluzione?
  - Comportamento degli utenti Spotify:
    - Quando la traccia corrente è a 30 secondi dal termine, i 92% degli utenti ascolterà la successiva nell'ordine previsto.
    - Se la traccia è a 10 secondi dalla fine, la percentuale sale al 94%
- Strategia finale:
  - 30 secondi al termine: avvia la ricerca di peers con il brano successivo
  - 10 secondi al termine: se nessun peer individuato, (scenario critico) scarica dal CDN

P2P – Spotify – 2008<->2014

- Cosa avviene se durante lo streaming tutti i peers interrompono l'upload?
  - Il client continuamente monitora il buffer disponibile. Se il buffer <3 secondi
  - EMERGENCY MODE
    Stop all'upload verso gli altri peers
    - Particolarmente uitle in caso di connessioni asimmetriche, dove la capacità di download viene ridotta in caso di uploads concorrenti (TCP!)
  - Download dal CDN Spotify

P2P – Spotify – 2008<->2014

- Le tracce venfono divise in **chunks** da 16KB
- Una traccia può essere scaricata in contemporanea dalla rete P2P e dalla CDN
- Se vengono utilizzate entrambe le reti, la CDN non viene mai utilizzata per più di 15 secondi
- Come selezioniamo i peers da cui scaricare I chunks?
  - Approccio greedy, basato sull'average download speed dei peers

P2P – Spotify – 2008<->2014

- Considerazioni:
- Spotify è un ottimo esempio di mix tra P2P e CDN
- Riesce a ridurre i costi (la banda non è gratis!) e a mantenere altre prestazioni
- Alcune scelte consentono di limitare I problemi tipici delle reti P2P (latenza, affidabilità)

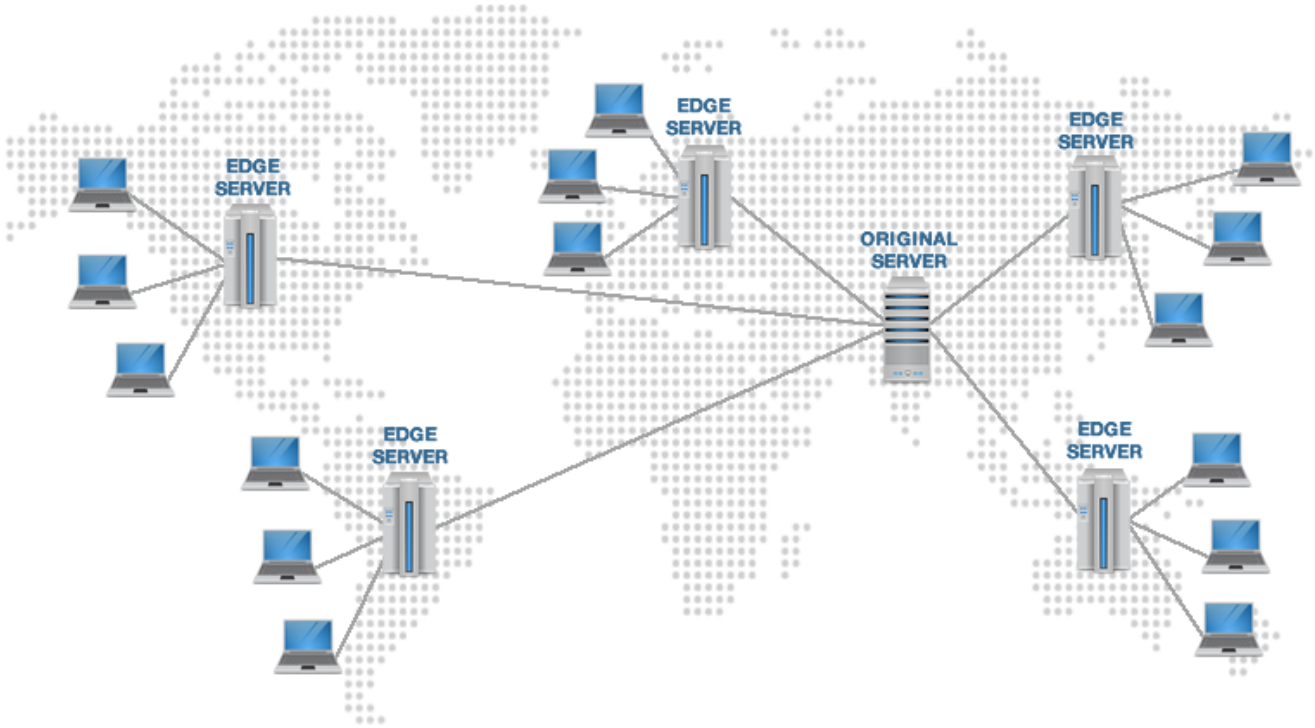
P2P – Spotify – 2014 - AWS

- Spotify interrompe l'utilizzo della rete P2P a favore di Amazon Cloud Front
- Perché?
  - Il numero di utenti mobile di Spotify cresce esponenzialmente (già prima nonin P2P!)
  - Architettura estremamente più semplice e IAAS
  - Stop all'utilizzo della banda degli utenti
  - Sistema più affidabile
- 2016: Spotify sposta parte dei servizi su Google Cloud
  - Perché?
  - Data Anaytics
  - Load Test: 2,000,000 di richieste per secondo

P2P – CDN

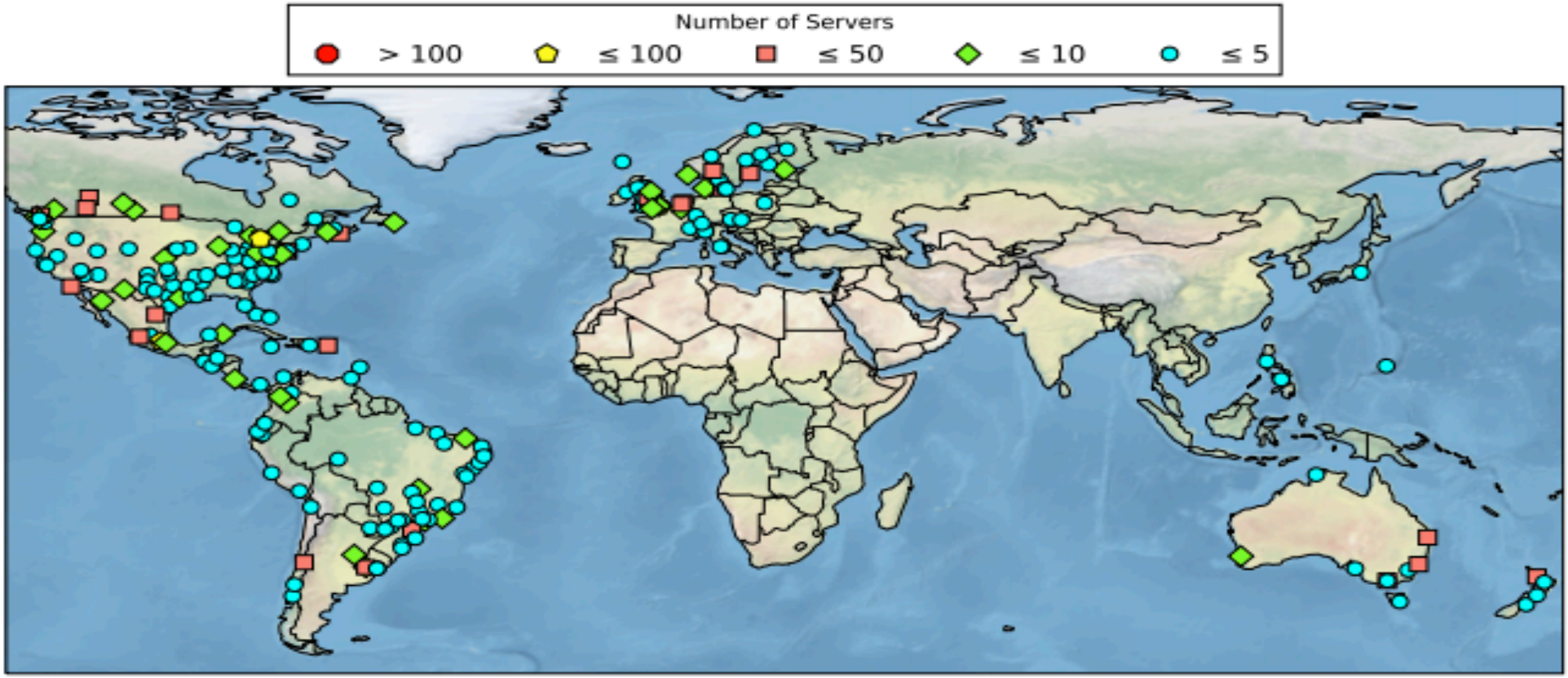
- Cos'è una CDN?
- Una rete di sistemi collegati che collaborano per la distribuzione di contenuti, in maniera trasparente per l'utente finale.
- Vantaggi Principali:
  - Ridurre il carico sui sistemi centrali
  - Velocizzare le connessioni negli endpoint più distanti
- Nel 1998 tre studenti dell'MIT partecipano ad una competizione per 50K$
- Nasce Akamai Technologies, la prima azienda nata specificamente come CDN
- Oggi serve: Adobe, Apple, Audi, Cathay Pacific, CBC, Cognos, Logitech, MTV Networks, Amazon.com, CBS, eBay, Facebook, FedEx, Microsoft, McAfee, NASA, NBA, NBC, NHL, NFL, Nintendo, IBM, IKEA, Reuters, Sony BMG, Symantec, U.S. Air Force, Yahoo!, la Casa Bianca

P2P – CDN



- Le richieste, invece di raggiungere il server centrale, intercettano un server edge, che può:
  - Rispondere raccogliendo informazioni dalla propria cache
  - Inoltrare la richiesta al server originale mediante una connessione ultrawideband

P2P – CDN Netflix



(b) CDN servers deployed within ISPs.

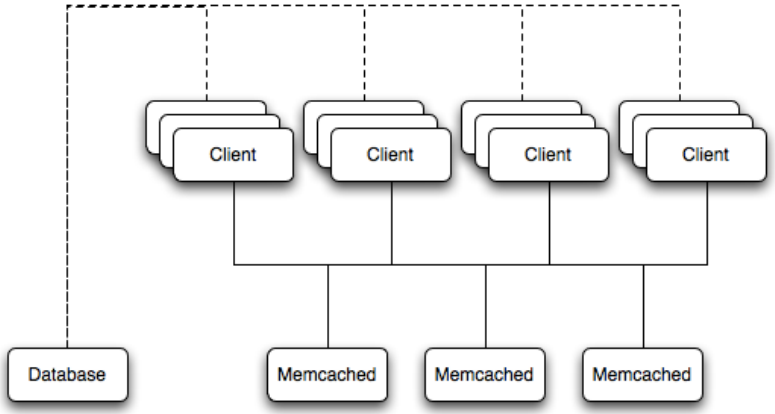
P2P – CDN: protezione DDOS

- DDOS: Distributed Denial of Service.
- Attacco svolto con l'obiettivo di rendere irraggiungibili alcune risorse di rete.
- Solitamente tenta di sovraccaricare alcune parti della rete, impedendone l'utilizzo agli utenti finali.

P2P – CDN: protezione DDOS

- Amplification Factor
  - Quando eseguiamo una richiesta di x byte ad un server, più è grande la risposta che potremmo ricevere maggiore è l'amplification factor
- Perché è importante per i DDOS?
  - L'attaccante, inviando pochi byte, può generare risposte da diversi megabyte
- Esempi di attacchi

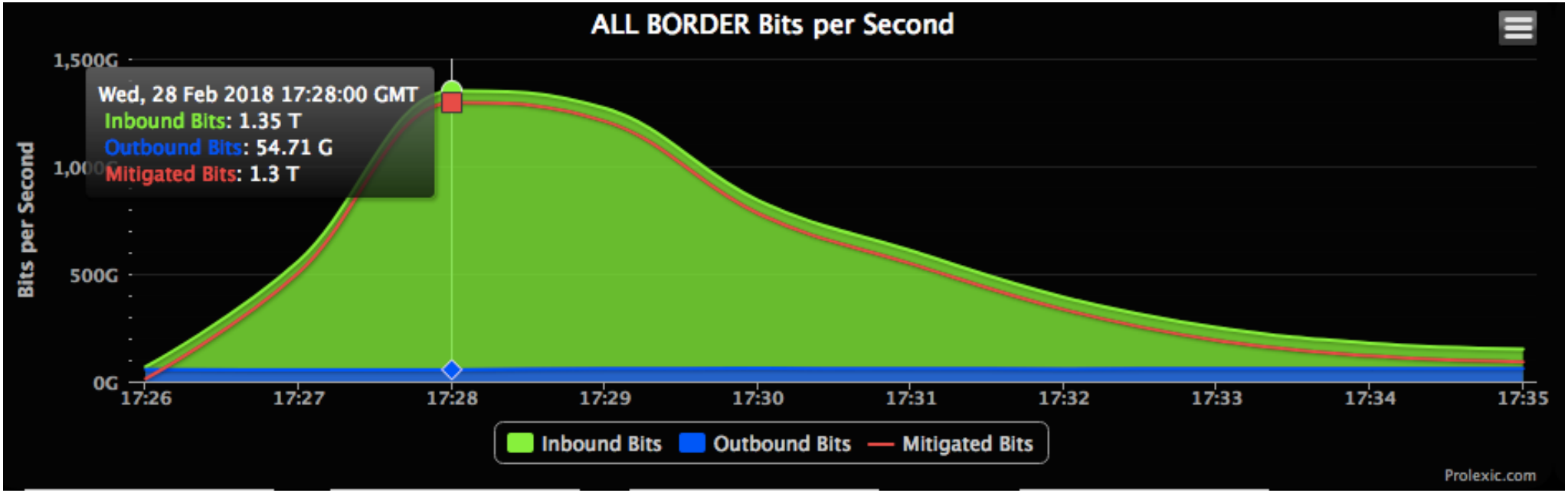
P2P – DDOS e memcached -> Github



- 80’000 istanze memcached aperte
- Amplification factor > 51,200x !

- 28/02/2018: il più grande attacco DDOS registrato nella storia:
- 1.35Tbps
- 126.9 milioni di pacchetti al secondo
- Github mitiga l’attacco in meno di 10 minuti. Come?

P2P – CDN come protezione DDOS



- Tutto il traffico di github viene dirottato sui sistemi akamai
- 1.3Tbps vengono filtrati dalla rete

CDN: amplification factor

- Un altro servizio?
- DNS
- Falsificazione dell’indirizzo IP sorgente e utilizzo del campo “ANY”
- TCP? UDP? o entrambi?
- Solo UDP!