Cenni di Applicazioni P2P

Reti di Calcolatori

Federica Paganelli

Paradigma peer-to-peer

- Finora applicazioni client-server
  - server offre servizio, client chiede di utilizzare servizio (HTTP, FTP, Telnet, SMTP, DNS)
- Paradigma P2P
  - tutti gli host sono *peer* e agiscono sia da client che da server

Problemi di copyright

Freenet
Napster
Gnutella
FastTrack (KaZaA)
Emule
BitTorrent
...

Applicazioni

- Distribuzione e memorizzazione di contenuti
  - Es. di file sharing: Gnutella, KaZaA, BitTorrent, eDonkey ed eMule, …
  - Es. di file storage: Freenet
  - Update di giochi, es. Blizzard (Diablo III, StarCraft II and World of Warcraft), Wargaming (World of Tanks, World of Warships, World of Warplane)
- Condivisione di risorse di calcolo (elaborazione distribuita)
  - Es.: SETI@home – Search for Extraterrestrial Intelligence
- Collaborazione e comunicazione
  - Es.: Chat/Irc, Instant Messaging, Jabber
- Telefonia
  - Es.: Skype
- Content Delivery Network
  - Es.: CoralCDN
- Piattaforme
  - Es:. JXTA (Piattaforma SUN dismessa intorno al 2010)
- Bitcoin

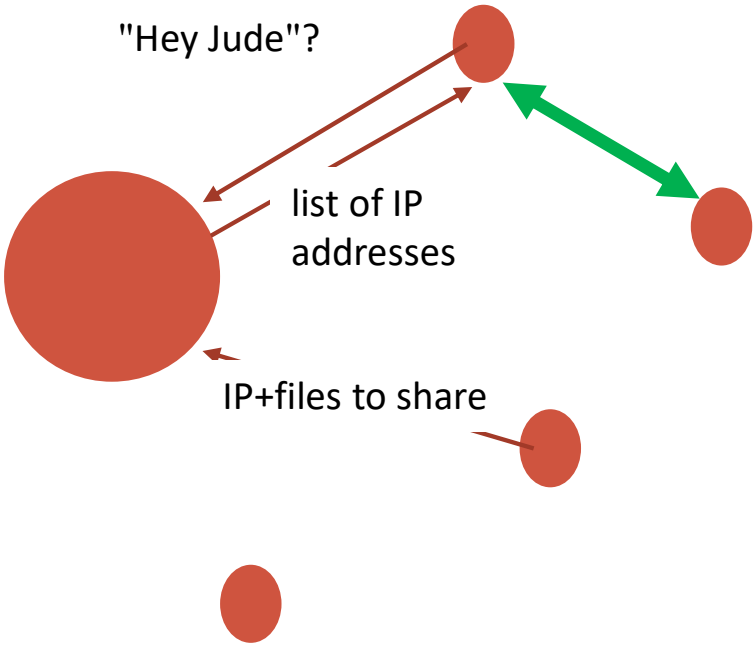
Peer

- Tutti i nodi (peer) hanno la stessa importanza (in linea di principio)
  - Nodi indipendenti (autonomi) e localizzati ai bordi (edge) di Internet
- Nessun controllo centralizzato
  - Ogni peer ha funzioni di client e server e condivide delle risorse
- servent = server + client
  - In realtà, possono essere presenti server centralizzati o nodi con funzionalità diverse rispetto agli altri (supernodi)
- Sistemi altamente distribuiti
  - Il numero di nodi può essere dell'ordine delle centinaia di migliaia
- Nodi altamente dinamici ed autonomi
  - Un nodo può entrare o uscire dalla rete P2P in ogni momento
- Operazioni di ingresso/uscita (join/leave) dalla rete anche sofisticate
  - Ridondanza delle informazioni

Problema

- Coppie di peer comunicano direttamente tra loro
- Ogni peer però
  - Non è necessariamente sempre attivo (connessioni intermittenti)
  - Può cambiare indirizzo IP ogni volta che si connette
- Come trovare i peer?
- Come tenere traccia dei file disponibili nei vari peer?

Directory centralizzata



- Directory -> client-server
  - Elenco peer e risorse condivise
- file transfer P2P
  - Napster

Problemi

- Unico punto di fallimento
- Collo di bottiglia per performance
- Facile da “oscurare”

Reti decentralizzate

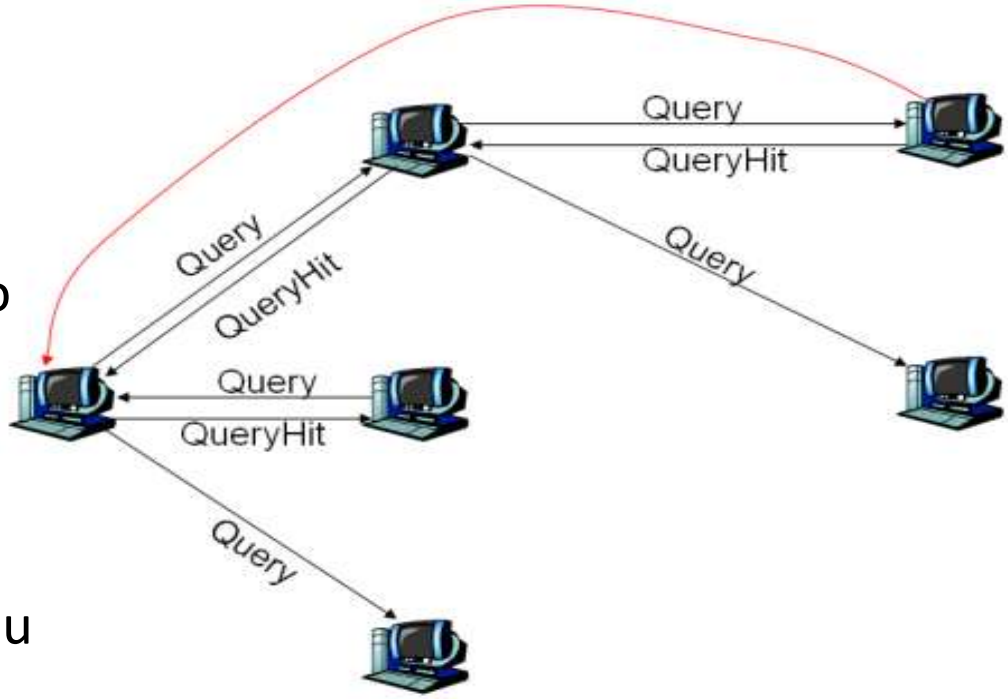
- Non c'è un servizio di directory centralizzato
- I peer si organizzano in una overlay network (rete logica)
- Due categorie:
  - Reti non strutturate
  - Reti strutturate

Reti non strutturate

- Nodi organizzati come un grafo in modo random
  - L'organizzazione della rete segue principi molto semplici
- Non ci sono vincoli sul posizionamento delle risorse rispetto alla topologia del grafo
  - La localizzazione delle risorse è resa difficoltosa dalla mancanza di organizzazione della rete
- L'aggiunta o la rimozione di nodi è un'operazione semplice e poco costosa
- Obiettivo: gestire nodi con comportamento fortemente transiente (tassi di join/leave elevati)
- Esempi: Gnutella, FastTrack, eDonkey/Overnet

Reti non strutturate - Query flooding

- Completamente distribuita (no server centralizzato)
- Esempio: Gnutella
  - Peer formano *overlay network*
    - archi=connessioni TCP
    - ogni nodo tipicamente connesso a 10 vicini
  - Peer invia query a tutti suoi i vicini
  - Se peer riceve query e ha il file richiesto invia messaggio *QueryHit* su reverse path, altrimenti inoltra la query a tutti i suoi vicini



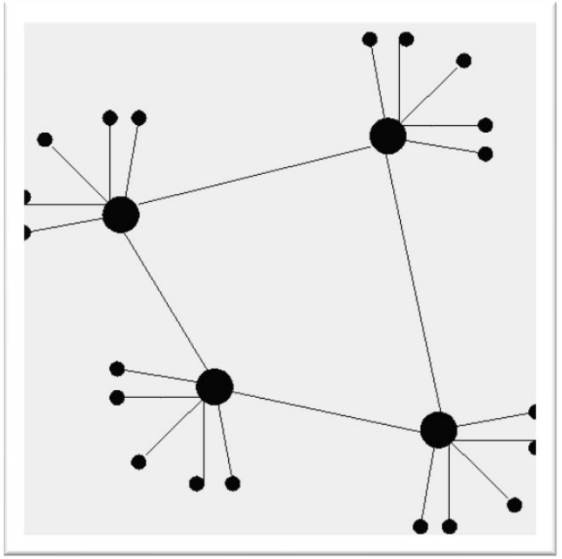
Problemi

- Flooding
- Poco scalabile

Bootstrap: il sw include un primo elenco di nodi

Copertura gerarchica

- Cerca di combinare il “meglio” dei due approcci precedenti
  - no server con tutti i contenuti (solo “bootstrap servers”)
  - i peer non sono tutti uguali (esistono i group leader)
- Ogni peer
  - è group leader (se “potente” in banda o risorse)
  - oppure viene assegnato a un group leader
- Connessioni TCP
  - tra peer e il suo group leader
  - tra (alcune) coppie di group leader

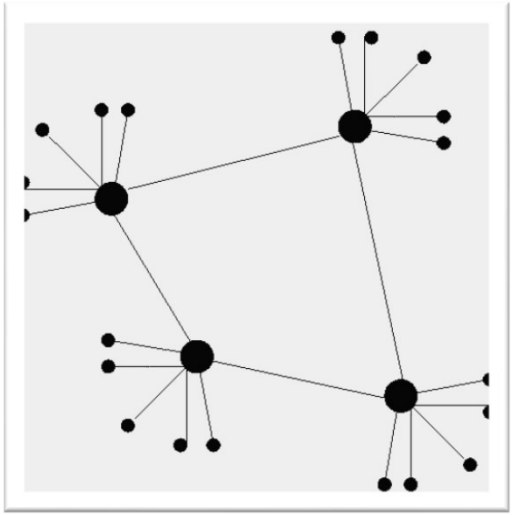


- Group leader tiene traccia del contenuto dei suoi “figli”

  Group leader ~ Napster-like mini-server

  Leader-to-leader connections ~ Gnutella-like overlay network

Copertura gerarchica



- Ogni file associato con un suo hash e un suo descrittore
  - Client invia una query di keyword a suo group leader
  - Group leader risponde con dei “match” del tipo

    < hash del file, indirizzo IP >
  - Se il leader inoltra la query ad altri leader, questi rispondono con altri match
  - Il cliente sceglie quindi i file da scaricare
- Protocollo per gestire disconnessione dei group leader: i peer di quel group leader devono essere assegnati ad un altro group leader

Reti strutturate - DHT

- Sistemi con Distributed Hash Table (DHT)
- Ad ogni peer è assegnato un ID ed ogni peer conosce un certo numero di peer
- Ad ogni risorsa condivisa (pubblicata) viene assegnato un ID, basato su una funzione hash applicata al contenuto della risorsa ed al suo nome
- Routing della risorsa pubblicata verso il peer che ha l'ID più «vicino» a quello della risorsa
- La richiesta per la risorsa specifica sarà instradata verso il peer che ha l'ID più «vicino» a quello della risorsa

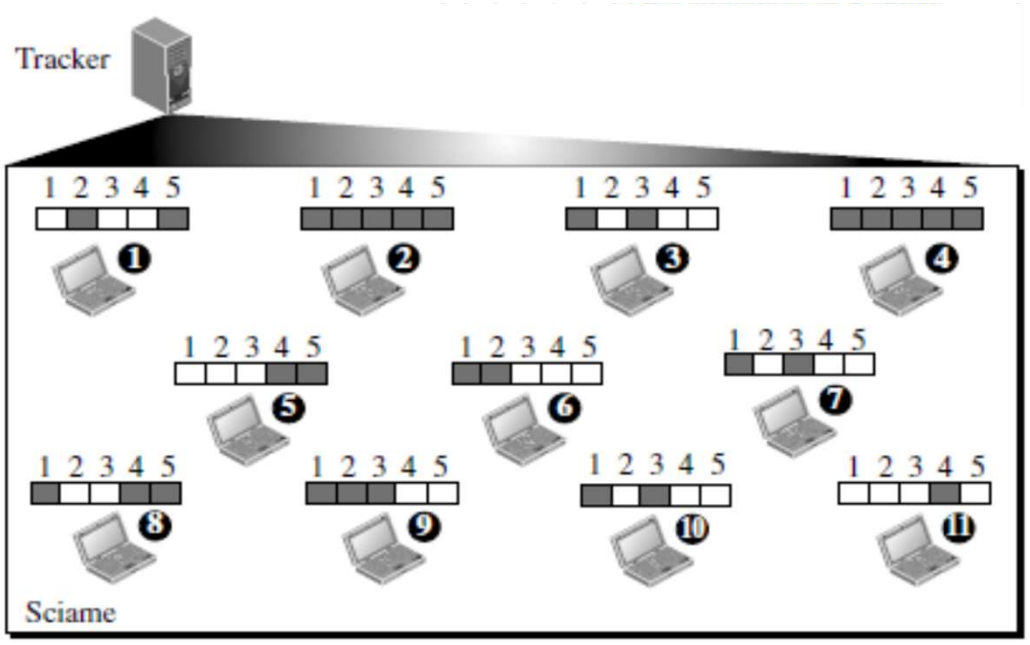
Reti strutturate

- Vincoli sul grafo (strutturato) e sul posizionamento delle risorse sui nodi del grafo
- L'organizzazione della rete segue principi rigidi
- L'aggiunta o la rimozione di nodi è un'operazione costosa
- Obiettivo: migliorare la localizzazione delle risorse
- Esempi: Chord, Pastry, CAN, alcune versioni di BitTorrent

BitTorrent

- Protocollo molto diffuso per la distribuzione di file in Interney
- Idea di base: dividere un file in pezzi (*chunk*) e far ridistribuire ad ogni peer i dati ricevuti, fornendoli a nuovi destinatari; in questo modo:
  - Si riduce il carico di ogni sorgente
  - Si riduce la dipendenza dal distributore originale
  - Si fornisce ridondanza
- Processo collaborativo di condivisione di un file: *torrent*
- insieme di peer che partecipano ad un torrent, i.e. alla distribuzione di un certo file, scambiandosi parti di file: *swarm*
- Ogni peer allo stesso tempo preleva e trasmette più chunk

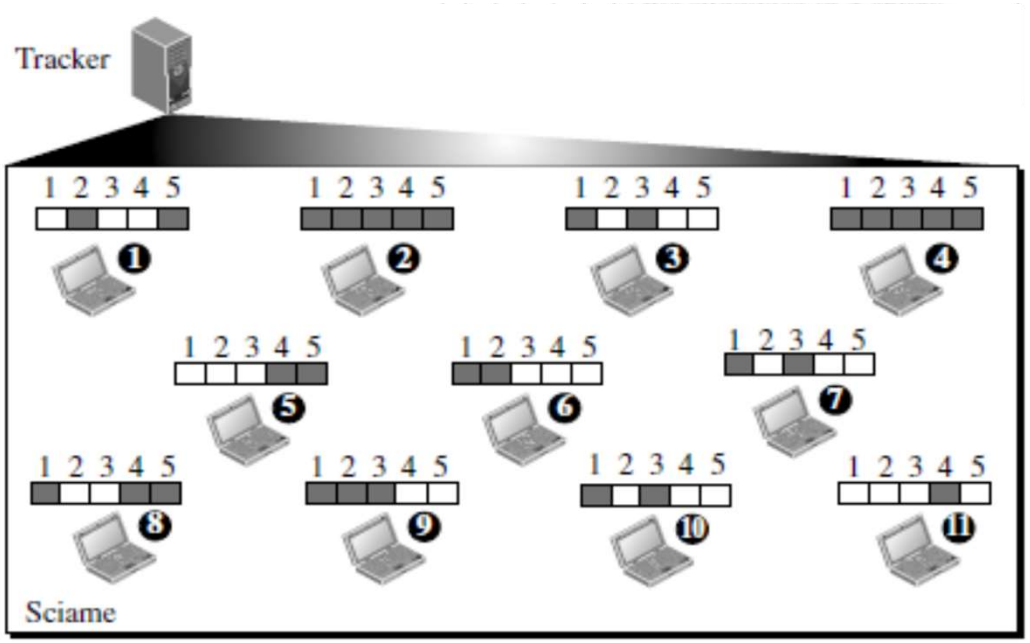
BitTorrent



- Tracker: il nodo che coordina la distribuzione del file
  - tiene traccia dei peer che stanno parte-cipando al torrent.
- Per condividere un file, un peer crea un file .torrent, che contiene metadati sul file condiviso e sul tracker

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

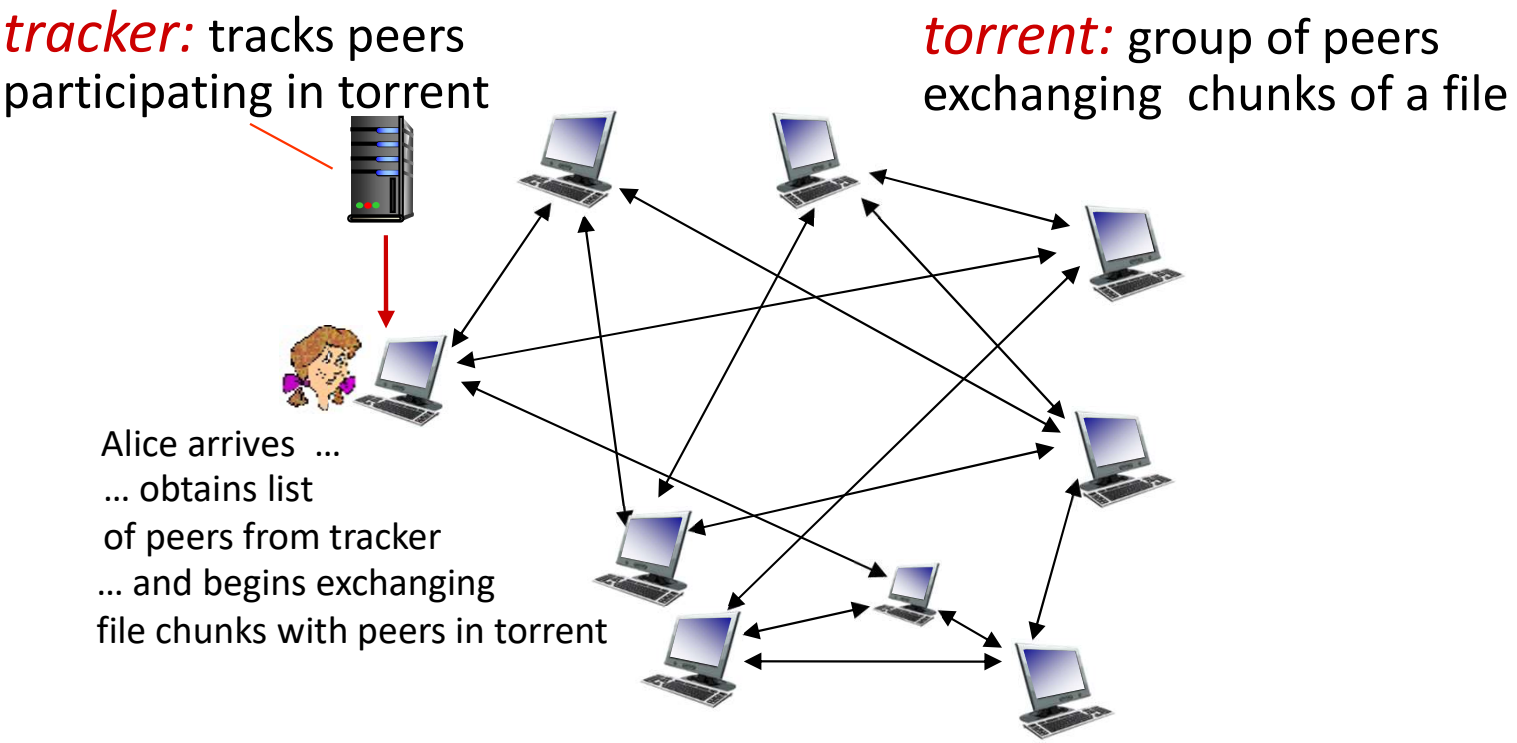
BitTorrent



- Nuovo peer
  - cerca (con motore ricerca) "Do what you u want" e ottiene un file *.torrent*, ovvero un metafile con info sui chunk e indirizzo IP di tracker
  - contatta tracker e riceve indirizzi di alcuni peer dello swarm

P2P file distribution: BitTorrent

- file diviso in chunk di 256Kb
- peers nel torrent inviano/ricevono chunk del file



BitTorrent

- Un peer, quando entra a far parte di un torrent per la prima volta, non ha chunk del file.
- Col passare del tempo accumula sempre più parti che, mentre scarica, invia agli altri peer.
- Una volta che un peer ha acquisito l'intero file, può (egoisticamente) lasciare il torrent o (altruisticamente) rimanere nel torrent e continuare a inviare chunk agli altri peer.
- Un peer può lasciare il torrent in qualsiasi momento con solo un sottoinsieme dei chunk del file e rientrare a far parte del torrent in seguito
- In un dato istante un peer avrà un sottoinsieme dei chunk del file e saprà quali chunk hanno i suoi vicini. Deve prendere due importanti decisioni:
  1. quali chunk deve richiedere per primi ai suoi vicini
  2. a quali vicini dovrebbe inviare i chunk a lei richiesti

BitTorrent

Strategie principali

1. “rarest (chunks) first” ($\rightarrow$ cosi diventano anche meno “rari”)

   Eccezione: in caso di download appena iniziato, Random First

2. "*tit for tat*" (pan per focaccia): inviare dati ai peer che inviano dati, scegliendo quelli che stanno inviando dati a frequenza maggiore

   - Ciascun peer invia dati ad al più 4 vicini
     - Priorità ai vicini che stanno inviando dati alla velocità più alta
   - Ogni peer classifica i suoi vicini in "soffocati"(choked) e «non soffocati"(unchoked)
     - aggiorna ogni 10 sec la sua classifica
     - ogni 30 sec sceglie un choked a caso e dopo uno scambio iniziale di file può promuoverlo (in questo modo i nuovi arrivati possono entrare)