P2P

In applicazioni con paradigma peer-to-peer, tutti gli host hanno la stessa importanza e agiscono sia da client che da server. Utilizzi: distribuzione di file, elaborazione distribuita, comunicazione testuale e audio/video...

Tipicamente le reti p2p sono altamente dinamiche, con ingresso e uscita continua di nodi gestita automaticamente, replicando i dati su più host. Per trovare i peer, che non sono sempre attivi e possono cambiare indirizzo IP ogni volta che si connettono, si possono adottare diverse strategie:

directory centralizzata i peer si registrano con il nodo directory, che consultano per trovare altri peer. Svantaggi: SPoF, bottleneck.

rete decentralizzata organizzazione dei peer in una rete logica (*overlay network*)

non strutturata grafo organizzato con regole semplici (sostanzialmente casuale), nessun vincolo sul posizionamento delle risorse rispetto alla topologia. Facile aggiungere/rimuovere nodi (utile con alti tassi di join/leave), ma localizzare le risorse è difficile.

Esempi:

query flooding (Gnutella) tipicamente ciascun host è connesso (TCP) a 10 peer, ai quali invia le query. Quando un nodo riceve una query, risponde con un messaggio *query hit* se ha il file, lo inoltra ai suoi vicini altrimenti. Problema: flooding, poco scalabile.

copertura gerarchica approccio misto tra la directory decentralizzata (peer–leader) e rete non strutturata (leader–leader):

- si individuano dei *group leader*, ogni peer è assegnato ad un group leader;
- le connessioni sono tra il peer e il suo group leader, e tra coppie (non tutte) di group leader;
- il leader tiene traccia del contenuto dei figli, e inoltra agli altri leader le richieste per file che non ha;
- si definisce un meccanismo per la riassegnazione dei peer in caso di disconnessione del leader.

A una richiesta di un file, il group leader risponde con match della forma $\langle$hash, indirizzo IP$\rangle$

strutturata vincoli rigidi sulla struttura del grafo, e.g. sistemi con DHT (distributed hash table):

- ogni peer ha un ID;
- ad ogni risorsa viene assegnato un ID calcolato con un hash del contenuto;
- la risorsa viene memorizzata dal peer con l'ID più vicino.

BitTorrent

I file sono divisi in pezzi (*chunk*), e ciascun peer ridistruibuisce i dati che riceve. La condivisione con *swarm* permette di ridurre il carico e la dipendenza dal distributore originale e garantire ridondanza.

Alcuni nodi sono *tracker* che coordinano la distribuzione dei file. Per scaricare dati, è necessario ottenere un file .torrent, che contiene metadati sul file e il tracker. Contattando il tracker si ricevono gli indirizzi di qualche peer.

Per lo scambio di dati:

- ciascun nodo è connesso ad al più 4 peer;
- per scoraggiare il leeching, si inviano dati ai peer che trasmettono di più;
- per permettere l'ingresso di nuovi nodi nello swarm, ogni 30 secondi si sceglie un peer *choked* (a cui non si stanno inviando dati) e si promuove;
- si cerca di scaricare per primi i chunk più rari.