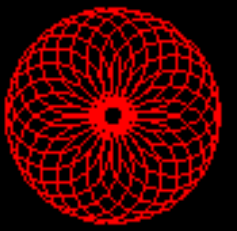
Lo Spirografo

www.spirografo.net



RETI DISTRIBUITE

Kademlia

Kademlia

- Kademlia e' un protocollo di rete *peer-to-peer* ideato da Petar Maymounkov e David Mazieres presso la New York University e rappresenta una delle molte versioni delle *tabelle di hash distribuiti*(DHT)
- Il protocollo di trasporto utilizzato e' l'UDP
  - Connectionless
  - Stateless
  - “Inaffidabile”...
  - Rapido ed efficiente

Kademlia

- Appartiene alla cosidetta “terza generazione” dei network *peer-to-peer*
  - Prima generazione: network, come Napster, che utilizzano un database centrale. Il trasferimento dei dati avviene da client a client. Particolarmente esposta ad attacchi di denial-of-service(DoS).
  - Seconda generazione: network, come Gnutella, che utilizzano la tecnica del flooding. Maggiore robustezza ad attacchi di DoS. Risulta pero' meno efficiente rispetto a Napster.
  - Terza generazione: network, come Freenet e Kademlia, che prevedono un meccanismo completamente distribuito.Questi network utilizzano le tabelle di hash distribuiti(DHT) e consentono di ottenere sia la decentralizzazione di Freenet e Gnutella sia l'efficienza di Napster.

Kademlia

- La rete Kademlia e' caratterizzata dalle seguenti tre costanti:
  - Alpha è un numero piccolo che rappresenta il grado di parallelismo nelle chiamate di rete, tipicamente vale **3**
  - B è la dimensione in bit delle chiavi usate per identificare i nodi ed i file messi in share; in Kademlia vale **160**
  - K è il massimo numero di contatti memorizzati nel bucket, normalmente è pari a 20
- Altre costanti che conviene citare sono:
  - tExpire = 86400 s (24 ore), il tempo dopo il quale la coppia chiave/valore muore, è il time-to-live (TTL) della data di pubblicazione originale.
  - tRefresh = 3600 s (un'ora), tempo dopo il quale il bucket deve essere aggiornato.

Kademlia

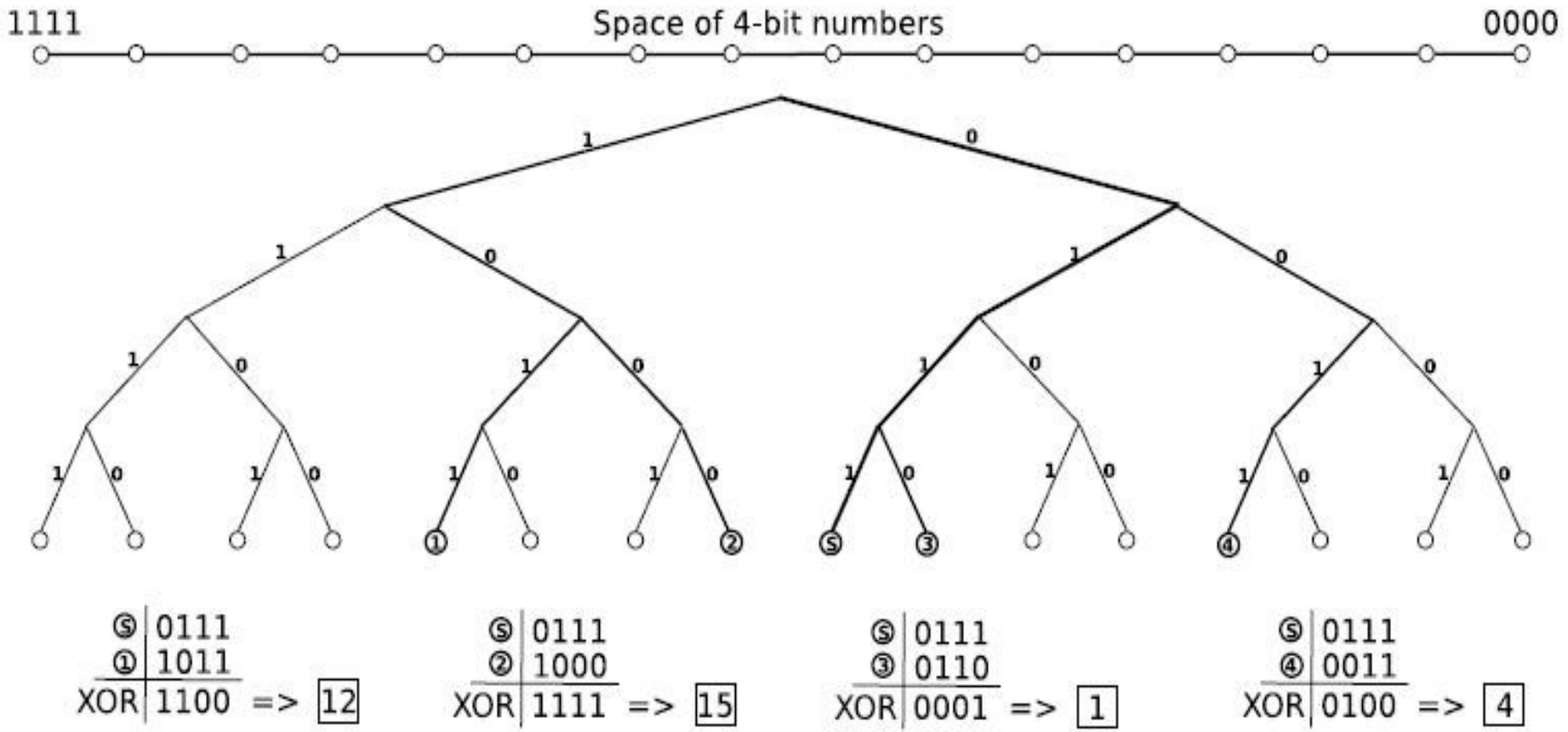
- Proprieta' di Kademlia
  - Decentralizzazione: realizza una rete completamente distribuita all'interno della quale i nodi comunicano tra di loro senza un coordinamento centrale
  - Scalabilita': il sistema garantisce un funzionamento efficiente anche in presenza di un numero molto elevato di nodi
  - Tolleranza ai guasti: il sistema non perde in efficacia anche se i nodi entrano od escono dalla rete, non e' vulnerabile ad attacchi di *denial-of-servie*(DoS) in quanto completamente decentralizzato e continua a funzionare correttamente anche in caso di rottura di alcuni nodi apparteneti al network

Kademlia

- Metrica utilizzata
  - La metrica utilizzata in Kademlia e' la "*XOR metric*". Le distanze tra due nodi nella rete Kad vengono calcolate utilizzando l'operatore logico XOR il quale restituisce vero se e solo se uno degli operandi e' vero, ma non entrambi
  - Alcune proprieta'
    - d(x,x) = 0
    - d(x,y) > 0 : x ≠ y
    - per ogni x, y : d(x,y) = d(y,x) ==> XOR e' *simmetrico*
    - proprieta' triangolare: d(x,y) + d(y,z) >= d(x,z)
    - XOR e' *unidirezionale*: dato un punto x e una distanza Δ > 0 esiste un solo punto y che soddisfa la seguente relazione: d(x,y) = Δ

Kademlia

- Un esempio



Kademlia

- NodeID
  - Stringa di 160 bit
  - Viene assegnato ad ogni nodo quando entra nella rete(*bootstrap*)
  - Identificativo del nodo
  - L'assegnamento del NodeID viene effettuato dal nodo stesso tramite una procedura quasi casuale ma con distribuzione uniforme, cio' nonostante la probabilita' che esistano due nodi con lo stesso ID e' pressoche' nulla
- FILEHASH
  - Stringa di 160 bit
  - Viene assegnato ad ogni file aggiunto al network
  - Identificativo del file
  - Non dipende dal nome del file, ma dalle sue dimensioni e contenuto

Kademlia

- Tabelle di hash distribuiti(1)
  - Le prime DHT sono state presentate nel 2001(CAN, Chord, Pastry, Tapestry)
  - Progettate per gestire reti formate da un numero elevato di nodi che si possono connettere e disconnettere con elevata frequenza(peer-to-peer)
  - Le proprieta' tipiche sono: *scalabilita*', *decentralizzazione* e *resistenza ai guasti, n*on a caso sono le stesse proprieta' di Kademlia che, come gia' detto, e' basato sulle DHT
  - E' inoltre richiesta una bassa dilazione(numero di passaggi per giungere a destinazione) ed un numero di vicini per ciascun nodo basso, cosi' da poter rendere minimo l'*hoverhead di mantenimento*

Kademlia

- Tabelle di hash distribuiti(2)
  - Costruita al di sopra di un *Keyspace* il quale e' formato da un set di stringhe di 160 bit
  - Ogni nodo e file all'interno della rete possiede una chiave univoca di 160 bit
  - Viene definita una funzione $d(key_1 , key_2)$ che prende il nome di *distanza* all'interno del *Keyspace* e che viene utilizzata per calcolare la “vicinanza” fra due nodi

Kademlia

- Tabelle di hash distribuiti(3)
  - Ogni nodo possiede delle liste contenenti i contatti a lui vicini(k-bucket)
  - Nei k-bucket per ogni 0 <= i < 160 ogni nodo tiene una lista di triplette (IP address, UDP port, NodeID) dei nodi ad una distanza compresa tra 2i e 2i+1 dal nodo stesso
  - Ciascun k-bucket risulta inoltre ordinato con il nodo visto meno di recente in testa e quello visto piu' di recente alla fine
  - Ciascun *k-bucket* viene aggiornato ogni volta che viene ricevuto un messaggio. La politica utilizzata e' quella di mantenere nella lista i nodi che sono attivi da maggior tempo: alcuni studi effettuati hanno infatti rivelato che maggiore e' il tempo che un nodo e' connesso alla rete e maggiore e' la probabilita' che questo rimanga connesso per un'altra ora

Kademlia

- Tabelle di hash distribuiti(4)
  - Aggiornamento dei k-bucket
    - Quando viene ricevuto un messaggio da parte di un altro nodo, se questo non e' presente nel k-bucket, ed il k-bucket non e' pieno, allora viene aggiunto alla fine della lista. In caso di k-bucket pieno il meccanismo e' differente: il nodo destinatario contatta il primo nodo della lista, se questo risponde, la lista rimane immutata, se invece non risponde allora viene eliminato dalla lista e viene aggiunto il mittente del pacchetto come ultimo contatto. Nel caso in cui il mittente del messaggio sia gia' presente nel k-bucket questo viene spostato ed inserito in testa alla lista.

Kademlia

- Tipologie di messaggi
  - PING: per verificare se un nodo e' ancora connesso
  - STORE: chiede di memorizzare una coppia (key, value)
  - FIND_NODE: contiene L'ID di un nodo come parametro. Il destinatario del messaggio ritorna (IP address, UDP port, nodeID) dei *k* nodi appartenenti al *k-bucket* piu' vicino. I valori contenuti nei pacchetti di risposta possono provenire anche da diversi *k-bucket* se quello piu' vicino non e' completo. In ogni caso quindi ritornano sempre *k* pacchetti. Se nella rete ci sono meno di *k* pacchetti verranno invece inviate le informazioni di ciascun nodo appartenete al network.
  - FIND_VALUE: si comporta come FIND_NODE, se pero' il nodo destinatario aveva ricevuto un messaggio di STORE per la chiave cercata questo ritorna solo il valore(*value*).

Kademlia

- Ingresso nella rete
  - Assegnamento NodeID
  - Il nodo che effettua il bootstrap deve conoscere indirizzo ip e porta UDP di un altro nodo appartenente alla rete(!)
  - Ottenuto l'indirizzo il nuovo nodo manda un messaggio di FIND_NODE avente come argomento il proprio NodeID(*self-lookup*), in questo modo vengono riempiti nuovi *k-bucket* con i nodi situati ad una distanza compresa fra il nuovo nodo ed il primo contattato

Kademlia

- Ricerca di un file
  - Il nodo manda FIND_VALUE(HASHFILE) a uno dei nodi nel k-bucket piu “vicino” al FILEHASH da cercare
  - Il nodo contattato risponde inviando l'indirizzo di un nodo piu' “vicino” al file
  - ...
  - Si giunge al nodo responsabile per quel file

Kademlia

- Aggiungere un file alla rete
  - Tramite FIND_NODE vengono localizzati i k nodi piu' “vicini” all'hash del file da inserire
  - Sono i nodi responsabili per quel valore/file
  - A ciascun nodo viene inviato il messaggio di STORE per quel valore

Kademlia

- Un paio di problemi
  - Possibilita' di eliminare dalla rete un valore/file. Chi attacca la rete infatti puo' fare in modo di ottenere NodeID tali da circondare il nodo responsabile per quel valore/file e non far passare alcun messaggio
  - Kademlia non ha nessuna difesa contro la possibilita' che all'interno della rete ci sia piu' di un nodo con lo setsso ID

Kademlia

- Software che utilizzano Kademlia
  - eMmule: dalla versione 0.40
  - aMule: dalla versione 2.1.0
  - MLdonkey: dalla versione 2.5-28
  - BitTorrent: dalla versione 4.1.0
  - µTorrent: dalla versione 1.2