

# PROTOCOLLO KADEMLIA



## ***Corso di Reti AA 2008/2009***

**Docente referente:** Simonetta Balsamo

**Studente:** Pesce Francesco

811366

fpesce@dsi.unive.it

# INDICE

1. Brevi cenni storici
2. Casi d'uso
  - 2.1. File sharing e generazioni
    - 2.1.1. Esempi pratici (rete Kad e Overnet)
3. Proprietà
4. Come funziona
  - 4.1. ID nodo e Filehash
  - 4.2. Tecnologia DHT > Kademlia > Rete KAD
  - 4.3. Protocollo di trasporto UDP
  - 4.4. Distanza nodi e metrica XOR
  - 4.5. Tabella di indirizzamento
    - 4.5.1. Aggiornamento tabella
  - 4.6. Messaggi
    - 4.6.1. Ping
    - 4.6.2. Store
    - 4.6.3. Find\_node
    - 4.6.4. Find\_value
  - 4.7. Operazioni
    - 4.7.1. Ingresso nella rete
    - 4.7.2. Ricerca di un file
    - 4.7.3. Aggiungere file in condivisione nella rete
    - 4.7.4. Uscita dalla rete
5. Vantaggi e svantaggi di utilizzo
6. Riferimenti

# 1. BREVI CENNI STORICI

Kademlia è un protocollo a livello network di rete peer-to-peer ideato da **Petar Maymounkov e David Mazières** della **New York University**, per un network di computer decentralizzato.

## 2. CASI D'USO

Le reti p2p hanno avuto una **costante evoluzione** (un'evoluzione spesso naturale, ma talvolta forzata) e oggi hanno raggiunto una larga diffusione e possono contare su un enorme numero di utenti a livello mondiale. Queste reti si possono classificare in base ad alcuni requisiti di funzionamento restringendoli a **tre generazioni**:

- PRIMA GENERAZIONE:

network, come Napster, che utilizzano un **database centrale**, ovvero **server che registrano le associazioni tra i client e i file che essi condividono**. Il trasferimento dei dati avviene da client a client. Particolarmente esposta ad attacchi di denial-of service (DoS), infatti una volta abbattuto il server principale la rete non può più funzionare. Di questa categoria facevano parte anche le prime versioni di Emule.

- SECONDA GENERAZIONE:

network, come Gnutella, che utilizzano la tecnica del flooding, ovvero una **rete completamente distribuita** (serverless), cioè dove **ogni nodo conosce i suoi vicini e a essi manda le richieste** che vengono propagate a caso per un numero massimo di salti. Ha una maggiore robustezza in caso di guasti ma risulta però meno efficiente rispetto a un network centralizzato, in quanto c'è lentezza di ricerca, overhead di comunicazione e possibilità di falsi negativi.

- TERZA GENERAZIONE:

network, come Chord, Freenet e Kademlia, che prevedono un meccanismo distribuito che è un **compromesso tra le prime due generazioni**. 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**, che per ora è il protocollo che ha avuto maggior successo, è usata da molti programmi di file sharing per la sua caratteristica di essere basata su tabelle distribuite su nodi della rete senza server centrali.

Tale protocollo è stato usato per la prima volta nell'ambito di un client per la condivisione di file, **Overnet**, del quale resta caratteristico, che è stato chiuso però per problema legali il 12 settembre 2005 dato che veniva utilizzato per scambi di file di grosse dimensioni (film, cd, dvd, ecc. ovviamente piratati). Tutt'ora la rete è ancora in piedi ma con limitata funzionalità.

Per evitare dannose sovrapposizioni tra le reti, rispettando la volontà dei puristi del protocollo Kademlia, gli sviluppatori di **eMule** hanno deciso di creare una **rete Kad del Mulo distinta** da quella di Overnet (la Medusa), che ha fatto la sua prima apparizione con le versioni (cosiddette "ibride": eD2k + Kad, appunto) 0.4x del Mulo.

Kad, come viene chiamata in gergo la rete serverless, è il futuro di eMule: non è ancora tempo per dire addio all'eredità di eDonkey, ma con l'introduzione dell'offuscamento anche per la rete distribuita, con il miglioramento della gestione dei flussi in entrata e uscita sulle porte dedicate, il protocollo si è ormai fatto abbastanza maturo per riuscire a garantire prestazioni, se non superiori, almeno equivalenti al precedente.

### 3. PROPRIETÀ

Kademlia è un protocollo P2P di overlay (ovvero che si poggia su protocolli di più basso livello ) progettato per reti decentralizzare peer to peer. Specifica la struttura della rete, regola la comunicazione tra i nodi e come deve aver luogo lo scambio di informazioni.

#### PROPRIETÀ DI KADEMLIA:

- **Decentralizzazione:** realizza una rete completamente distribuita all'interno della quale i nodi comunicano tra di loro senza un coordinamento centrale. Su Kademlia possono essere effettuate ricerche per parole chiavi per reperire il file ricercato e il compito di memorizzare l'indice dei file esistenti viene diviso paritariamente tra tutti i client.
- **Scalabilità:** 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-service(DoS) in quanto completamente decentralizzato e continua a funzionare correttamente anche in caso di rottura di alcuni nodi appartenenti al network.

#### LE CONDIZIONI SU CUI LA RETE DEVE RIUSCIRE AD ESSERE OPERATIVA SONO:

- **Instabilità dei nodi**, infatti i nodi si possono disconnettere senza preavviso, in media nel giro di un'ora solo la metà dei nodi presenti all'inizio sono ancora contattabili, e contemporaneamente comunque nuovi nodi si aggiungono.
- **Ogni nodo ha una capacità diversa** nel fare il suo dovere (limitatezza di banda, capacità di memoria, ecc..).

Gli unici **requisiti** del protocollo Kademlia sono che il computer connesso sia identificato tramite **IP** e che vi sia a disposizione una **porta UDP** aperta riservata (di solito 4672).

## 4. COME FUNZIONA

DHT > KADEMLIA > KAD

**Molti pensano che kad, la rete usata da emule, sia il diminutivo di kademia, e che siano quindi la stessa cosa. Ma non è così, almeno non proprio.**

La visione più teorica di questo network è la DHT, Distribuite Hash Table, dalla quale discende Kademia, la quale fissa alcuni punti chiave, che è a sua volta padre della rete Kad, la quale è la parte pratica e viene utilizzata sul client Emule.

- **DHT:** questa tecnologia è nata proprio sulle esigenze dei programmi di scambio file su rete p2p ed è il punto di incontro tra una rete centralizzata e una completamente distribuita. E' da questa infatti che Kademia ricava le sue proprietà di **decentralizzazione, robustezza e scalabilità**. Il Keyspace, ovvero il set di stringhe univoche corrispondenti a file (ID), viene partizionato tra tutti i nodi, assegnando a ogni nodo una set di chiavi di cui ne diventa il responsabile. L'Overlay network organizza in modo strutturato la rete, ovvero dato che ogni nodo conosce i suoi "vicini", e si riesce così a ricreare una topologia di rete. I messaggi di cui ogni nodo dispone sono quello di put(k), che fa conoscere agli altri nodi responsabili quali file il nodo mittente condivide, e il messaggio di get(k), che serve a far sapere al nodo mittente chi possiede tale file k condiviso inoltrando il messaggio attraverso l'overlay verso il nodo responsabile di k. Ogni nodo si connette alle rete contattando un nodo qualsiasi già dentro (bootstrap), diventando così responsabile di un determinato set di chiavi. Un nodo può disconnettersi dalla rete in qualsiasi momento in modo inaspettato oppure creando ridondanza dei valori di cui era responsabile per poi sconnettersi senza danni alla rete.
- **Kademia:** questo protocollo si basa fortemente sulla tecnologia DHT e definisce alcuni concetti lasciati astratti dalla tecnologia madre. In particolare fissa la determinazione degli id, la metrica per calcolare la distanza tra i nodi, introduce la tabella degli indirizzi che ogni nodo possiede per ricordarsi dei vicini e ulteriori tipi di messaggi tra i nodi. Vedremo comunque meglio tutti questi punti.
- **Rete KAD:** è una rete **basata completamente sul protocollo Kademia, ma con alcune leggere modifiche**. Infatti possiamo trovare l'introduzione dei buddy peer, ovvero dei compagni che aiutano il nodo connesso con problemi di porta UDP. Come per Kademia sulla DHT non vengono pubblicati i file o parti di file ma solamente i riferimenti al file. Possiamo trovare però due tipi di riferimenti: le fonti (ID del file, locazione del peer che lo condivide) e i metadati (parole chiave, ID dei files). Le fonti sono riferimenti di primo livello e sono esattamente come li definisce la rete Kademia, mentre i metadati sono riferimenti di secondo livello, sono una novità rispetto alla rete Kademia, e sono dei puntatori logici a più riferimenti di primo livello dato che ad una parola chiave possono corrispondere più files. L'ulteriore problema che Kademia non prevede è che lo stesso file (o pezzo del file) può essere condiviso da più utenti. Per non avere un Overload è stata aggiunta una protezione e qualsiasi peer kad può memorizzare al massimo 50000 riferimenti

diversi per un determinato fileID e in totale mantiene salvati fino a 60000 riferimenti come somma per tutti i fileID. Sempre per sicurezza ogni peer mantiene salvate fino a 300 fonti per ogni file, eliminando le meno recenti, come nella tabella degli indirizzi prevista da Kademlia, e aggiornando i riferimenti periodicamente proprio come previsto nelle DHT.

Le ricerche avvengono in 2 passi, utilizzando sempre la prassi introdotta da Kademlia con una strategia di routing che sfrutta la iterazione e il parallelismo (da vedersi poi in dettaglio). Nel primo passo si fa una ricerca della parola chiave tra i riferimenti di metadati e poi nel secondo passo per i file messi nella lista di download si fa una ricerca tra i riferimenti di fonti. Per il download si utilizza però un protocollo di trasporto TCP.

Esiste una rete alternativa KADu utilizzata da utenti Fastweb perché questo provider non è direttamente connesso al web, ma è una sorta di LAN, con conseguenti problemi di comunicazione.

## PROTOCOLLO DI TRASPORTO UDP

**UDP** (User Datagram Protocol): è un protocollo **non connesso, non affidabile e non confermato**, i pacchetti possono arrivare in ordine diverso o non arrivare affatto.

Quindi ricapitolando:

- si mandano frame indipendenti
- i frame non vengono confermati
- non si stabilisce una connessione
- i frame persi non si recuperano (in questo livello)

**Ma dalla sua ha la proprietà di essere estremamente veloce e efficiente.**

## NODOID E FILEHASH

Ogni nodo che si connette a tale rete deve potersi riconoscere e a tale scopo gli viene assegnato un **ID univoco** (di solito lungo **160 bit**).

Stesso vale per ogni singolo file condiviso, il programma client identifica i file in modo univoco nel network tramite il **filehash** (con algoritmi SHA, Secure Hash Algorithm) che il programma stesso assegna al file non appena questo è posto dall'utente in condivisione. Questo permette al network di riconoscere con certezza l'identità di un file. Anche perché questo **NON dipende nella maniera più assoluta dal nome del file**, ma dalle sue dimensioni e contenuto. Con questi presupposti, la possibilità che esistano nel network due file con identico hash è praticamente impossibile (se non tecnicamente, statisticamente lo è).



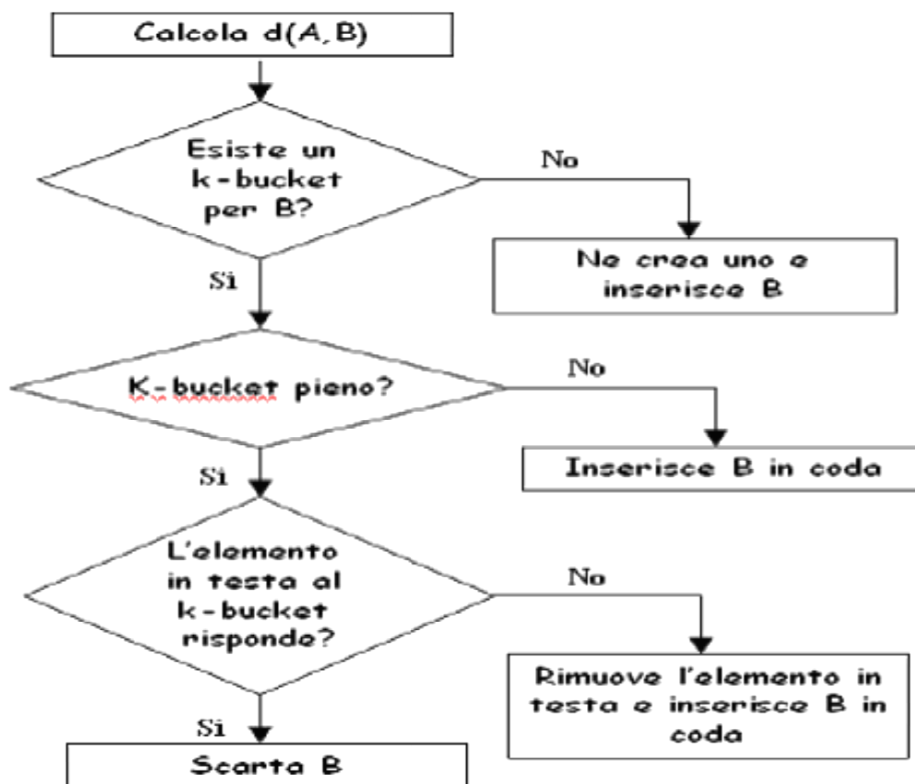
## TABELLA DI INDIRIZZAMENTO

**Ogni client (ogni singolo nodo) si crea una Tabella di indirizzamento** (Routing Table, su emule viene salvata nel file nodes.dat). La tabella consiste in **un array**, grande quanti i bit della lunghezza del nodo (solitamente una stringa da **160 bit**), **di liste** (k-bucket, solitamente al massimo 20 contatti) **di nodi**, di cui si salva la tripletta di informazioni a riguardo (**IP, porta UDP, ID nodo**). **Ogni k-esima lista comprende i nodi**, che si sono **incontrati nelle operazioni** di ricerca, memorizzazione e anche nel corso di attività aiuto alla ricerca per conto di un altro nodo, **che possiedono una distanza pari a k**. Con molta probabilità le liste alle prime posizioni saranno le più popolate e man mano sempre meno poiché il numero dei possibili ID nodo è molto maggiore di ogni possibile popolazione di nodi appartenenti alla rete.

È statisticamente noto che i nodi che sono collegati alla rete da più tempo hanno maggiori probabilità di rimanere collegati per un maggior tempo nel futuro. A causa di questa distribuzione statistica, Kademia seleziona i nodi connessi da maggior tempo per essere memorizzati in un k-bucket. Ciò aumenta la conoscenza di nodi validi nel futuro e rende più stabile la rete.

**Questa tabella** in caso di non venga compiuta alcuna azione (ricerche, inserimento, ecc) **dovrà essere aggiornata** (mediamente ogni ora) per non avere una visione della rete errata che comporterebbe grosse perdite di tempo in fase di ricerca e infatti ogni nodo ripubblica ogni ora i file condivisi.

Normalmente però non mancano continue ricerche di fonti (dato il principale utilizzo su programmi di scambio dati) e quindi **per ogni nodo contattato che si vuole salvare si attuerà il seguente algoritmo:**



Kademlia offre **quattro modalità di contatto tra i nodi**:

- **PING** - utilizzato per **verificare** che un nodo sia ancora presente e attivo.
- **STORE** (Chiave, Valore) - utilizzato per archiviare un **riferimento (coppia chiave-valore)** nel nodo o nei nodi più vicini alla chiave che ne farà da responsabile.
- **FIND\_NODE** (ID) - utilizzato **per cercare nodi, data una chiave** (ID di un nodo o di un file) **ritorna** una tripletta di informazioni (**IP, porta, NodeID**) **dei nodi contenuti nel k-bucket più vicino alla chiave ricercata del nodo contattato**.
- **FIND\_VALUE** (ID) - come FIND\_NODE, ma se il ricevente dispone della chiave richiesta, **se ne è cioè il responsabile**, al posto di ritornare la tripletta di ulteriori nodi più vicini (probabilmente lui è già il più vicino) **fornisce il valore corrispondente alla chiave**.

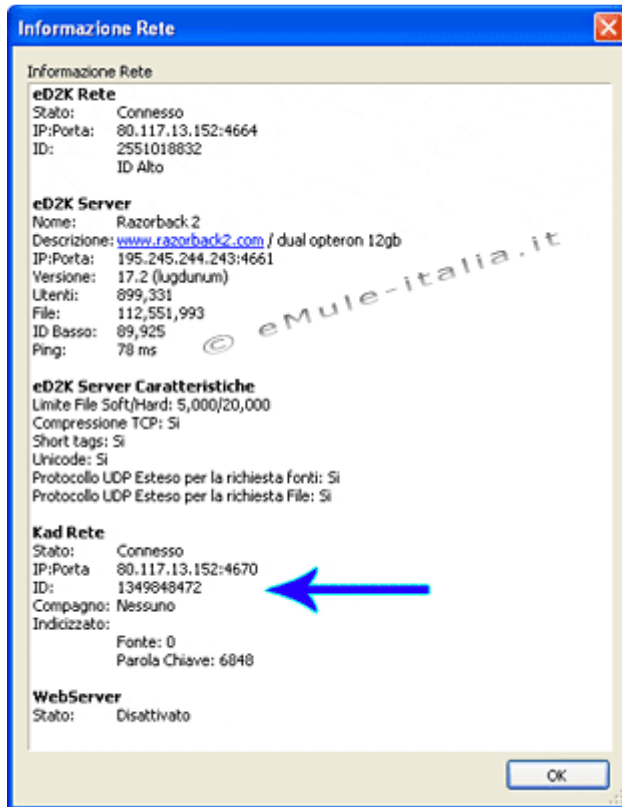
Ogni messaggio tra i nodi include un valore random generato dall'iniziatore della richiesta. Tale valore è utilizzato per assicurare che la risposta sia relativa alla domanda effettuata.

Find\_Node e Find\_Value, sebbene abbiamo funzioni diverse, lavorano nel medesimo modo. L'algoritmo (di prefix-matching) che utilizza questi messaggi funziona così:

- Il nodo sorgente vuole cercare il nodo obiettivo, molto facilmente non lo conosce (ovvero non è presente nei suoi k-bucket), così il nodo sorgente contatta il nodo più "vicino" che conosce al nodo obiettivo.
- Questo nodo "intermediario" se conosce il nodo obiettivo ne restituisce l'indirizzo, o, se nemmeno lui lo conosce, un ulteriore nodo da contattare che sarà il nodo che conosce più "vicino" al nodo obiettivo; il passaggio precedente continuerà iterativamente eliminando a ogni salto  $\frac{1}{2}$  dei nodi candidati (inizialmente tutti quelli della rete).
- Alla fine passando per vari nodi intermediari (vari hop, salti) l'ultimo nodo intermediario contattato fornirà finalmente l'indirizzo del nodo obiettivo.

La rete supporta tali operazioni:

- **INGRESSO NELLA RETE**



La sola **cosa necessaria per connettersi a questa rete è l'IP e la porta UDP di un qualsiasi client già connesso**. Questa operazione si chiama **Boot Strap**.

**Per l'assegnazione dell'ID il client genera un numero casuale all'interno del set di chiavi di cui sarà responsabile**, cioè farà da server per i file di cui avrà salvato la coppia Key/Value. Poi effettuerà l'operazione di **FIND\_NODE**, **avente come argomento il proprio NodeID (selflookup)** e così il nodo si farà conoscere dai nodi contattati e viceversa verranno riempiti nuovi k-bucket del nuovo nodo con i nodi contattati.

Una funzione aggiuntiva su Emule elimina anche il problema della necessità della porta UDP. Infatti una volta che il client è sulla rete chiede agli altri client di determinare se può essere contattato senza problemi. Se si ti verrà dato lo stato open, se invece non sei liberamente raggiungibile ti verrà assegnato lo stato firewalled. Dalla versione 0.44a di eMule in poi, la rete Kad supporta una funzione 'Compagno' per gli utenti firewalled. Compagni sono altri utenti connessi alla rete Kad che hanno lo stato Open e fungono da tramite per le connessioni che gli utenti firewalled non possono gestire.

- **INSERIMENTO FILE**

Ho un file che voglio inserire nella rete, ci calcolo l'hash. **Attraverso il FIND\_NODE trovo i nodi o il nodo che abbiano il Nodoid più simile ("vicini") all'hash del file da inserire.** A ciascun nodo trovato **mando un messaggio di STORE facendoli diventare responsabili per quel valore/file.** Essi così **potranno reindirizzare a noi chi cerca un certo file da noi condiviso.**

Per proprietà della DHT la rete (i nodi responsabili) memorizza coppie del tipo (chiave, valore).

- **Chiave:** Hash(file "aaa.xxx")=010101000100100
- **Valore:** riferimento al dato ricercato (es: indirizzo fisico del nodo che memorizza il contenuto)

**Oltre alla coppia chiave/valore il nodo responsabile si salva anche l'indirizzo IP, la porta UDP e l'ID** del nodo che fisicamente possiede il file in condivisione per poter poi restituire tale valore al nodo che lo cercherà tramite la richiesta FIND\_VALUE.

Può capitare che questo nodo responsabile noti un nodo ancora più vicino all'hash del file, probabilmente appena connesso alla rete, e allora vi ci duplica la coppia chiave/valore. In questo modo si creerà una replicazione dei dati che consentirà di trovare il file cercato anche nel caso della caduta improvvisa di un nodo.

Se una chiave comincia a essere popolare questa duplicazione avverrà anche su altri nodi vicini in maniera di velocizzare le ricerche (effettuo una sorta di caching).

- **RICERCA FILE**

Non esistono server che raccolgono tutti i file condivisi dei client in un database. In poche parole **ogni client è un piccolo server.** Ogni client è identificato da un valore di hash univoco, Kademia associa una certa "responsabilità" basata su questo hash. Ogni client connesso alla rete Kad agisce come un server per certe parole o fonti. L'hash del client determina la parola o le fonti specifiche ed in tal modo **qualunque tipo di ricerca diventa trovare quei client che hanno responsabilità relativa all'argomento della ricerca in corso.**

In Kademia **la tabella di routing di ogni nodo può essere rappresentata come un albero binario non bilanciato.**

L'algoritmo utilizzato da Kademia (di **prefix-matching**) procede iterativamente nella ricerca delle chiavi attraverso la rete, avvicinandosi di un bit al risultato ad ogni passo compiuto. Ne consegue che **il costo dell'operazione sia  $O(\log(n))$**  e che quindi **in una rete Kademia con  $2^n$  nodi, richiederà al massimo  $n$  passi per trovare il nodo cercato.**

– Il nodo manda FIND\_VALUE(HASHFILE) a uno dei nodi (o più di uno in maniera parallela) nel kbucket più "vicino" al FILEHASH da cercare

– Il nodo contattato risponde inviando l'indirizzo di un nodo (o più nodi per parallelismo) più "vicino" al file

– ...il nodo che inizia il lookup si salva gli id ritornati e si ritorna al passo 1...

– Si giunge al nodo responsabile per quel file che fornirà l'IP e porta di chi fisicamente possiede il file

E' da ricordare che ci sono più strade percorribili per raggiungere un nodo e per questo le ricerche vengono parallelizzate. Da questo ne consegue un'alta efficienza di ricerca con un basso costo (essendo distribuito).

- **USCITA DALLA RETE**

- **Ritiro Volontario di un nodo**

- Partizionamento della propria porzione degli indirizzi sui nodi vicini
    - Copia delle coppie chiave/valore sui nodi corrispondenti
    - Eliminazione del nodo dalle tabelle di routing

- **Fallimento di un Nodo**

- Se un nodo si disconnette in modo inatteso, tutti i dati memorizzati vengono persi a meno che non siano memorizzati su altri nodi (memorizzazione di informazioni ridondanti/replicazione)
    - **Perdita delle informazioni**, è necessario refreshing periodico delle informazioni
    - **Probing periodico** dei nodi vicini per verificarne la operatività con il messaggio di PING. In caso di fault, aggiornamento della tabella degli indirizzi per utilizzare percorsi di routing alternativi/ridondanti

## 5. VANTAGGI E SVANTAGGI DI UTILIZZO

### VANTAGGI:

- Per la struttura decentralizzata che chiaramente aumenta la **resistenza contro attacchi denial of service o guasti**. Anche se un intero insieme di nodi sono inutilizzabili si avranno effetti molto limitati sulla rete, che supererà automaticamente il problema isolando la rete intorno a questi nodi problematici.
- Sempre grazie alla decentralizzazione **si evitano i colli di bottiglia**.
- **Realizzazione semplice ed efficiente, indipendentemente dal numero dei nodi connessi**.

### PROBLEMI:

- Possibilità di eliminare o **perdere dalla rete un valore/file** in caso di fallimento di un nodo. Chi attacca la rete infatti può fare in modo di ottenere NodeID tali da circondare il nodo responsabile per quel valore/file e non far passare alcun messaggio.
- Kademia non ha nessuna difesa contro la **possibilità che all'interno della rete ci sia più di un nodo con lo stesso ID**, anche se estremamente difficile è possibile.
- Le **query per la ricerca dei file sono estremamente semplici** e infatti la rete non supporta query più complesse.

### DAL PUNTO DI VISTA PRATICO:

Occorre ricordare che con la Kad si possono ottenere risultati del tutto simili a quelli ottenuti connettendosi con i Server ma esistono alcune differenze sostanziali:

**DOWNLOAD:** la Kad **impiega un po' più tempo per ricercare le fonti** (con molta approssimazione indicativamente si può quantificare in 40/50 minuti circa il "maggior ritardo") **dato che dobbiamo aspettare il riempimento dei nostri k-bucket**

**UPLOAD:** i **file che condividiamo impiegano** ugualmente po' più **tempo per essere condivisi** (ritardo simile a quello in download), aspettando **il refresh della rete**

**RICERCHE:** la ricerca è effettuata su tutta la rete Kademia e non si ottengono risultati dai server, inoltre deve sempre cominciare con una parola di 3 o più lettere. Le ricerche su Kademia devono sempre cominciare con una parola di 3 o più lettere. Dopo la prima parola non ci devono essere altri operatori se non AND. Forse perchè ogni ricerca con Kad aumenta leggermente l'overhead e l'occupazione di banda (per le ricerche appunto). Se si dovesse cercare con soli 2 caratteri si avrebbero "una marea" di risultati e quindi una maggiore "congestione" della rete..

Rispetto a quella effettuata con i Server è **leggermente più lenta e in alcuni casi**

**meno precisa** (possibilità maggiori di trovare file rinominati), d'altro canto la ricerca effettuata su Kad offre spesso maggiori risultati e questo può essere molto utile per cercare file rari.

## 6. RIFERIMENTI:

- **Kademlia: A Peer-to-Peer Information System Based on the XOR Metric** di *Petar Maymounkov and David Mazières*  
<http://pdos.csail.mit.edu/~petar/papers/maymounkov-kademlia-lncs.pdf>
- **Guida a Kademlia** di *Emule.it*  
[http://www.emule.it/guida\\_emule/guide/guida\\_kademlia.asp](http://www.emule.it/guida_emule/guide/guida_kademlia.asp)
- **La rete Kademlia** di *Emule-italia.it*  
[http://www.emule-italia.it/emule\\_rete\\_kademlia.html](http://www.emule-italia.it/emule_rete_kademlia.html)
- **Lucidi del corso di Peer to Peer** di *Laura Ricci*  
<http://www.cli.di.unipi.it/doku/doku.php/p2p/start>