Lezione n.7
EMULE:
IL SISTEMA SERVER-BASED

Laura Ricci
24/10/2013

EMule: UN PO' DI STORIA....

- Origine: il protocollo eDonkey, il client edonkey2000 (ed2k)
- E-donkey: rete peer-to-peer per la condivisione di file di qualsiasi tipo
- I client eDonkey si connettono alla rete per condividere file.
- I server eDonkey hanno una funzione simile al server NAPSTER: permettono agli utenti di localizzare i file all'interno della rete.
- Aggiunta dinamica di servers alla rete.
- A causa del costante cambiamento della rete dei server, i client devono aggiornarne costantemente la lista.
- Ultime versioni: utilizzano la DHT Kademlia

EMule: UN PO' DI STORIA.....

- eMule: il nome del progetto sottolinea sia le somiglianze che le diversità con il programma eDonkey2000
- eMule (mulo) simile ad un donkey (asino). Emule "emula" le funzionalità di eDonkey.
- sviluppato nel maggio 2002 dal programmatore tedesco Merkur, come un programma eDonkey-compatibile ma con molte più funzioni.
- sfrutta due diversi tipi di overlay networks
  - La rete eDonkey (ed2k), basata sul modello client/server.
  - La rete KAD (presente a partire dalla versione 0.42)
    - implementa KADEMLIA, una DHT molto efficiente
    - rete serverless in cui la gestione dell'indice è distribuita a tutti i client connessi invece che essere affidata ad un insieme di server
- Utilizzato ogni giorno in media da 3 milioni di utenti che condividono circa 500 milioni difiles

EMule: LA RETE ED2K

- Sistema P2P orientato al file sharing
  - basata su una estensione del protocollo eDonkey
  - architettura Client/Server
  - utilizza sia il protocollo TCP che il protocollo UDP
- Client open source: miglioramenti continue delle versioni del client eMule
- Il protocollo definisce le interazioni tra
  - Client e server
  - Client e client
- Strategie originali implementate nel client
  - Sistema dei crediti
  - Gestione delle code
  - Gestione (recovery) di parti di file corrotte: corruption handling

eMule: IL CLIENT

Ogni client eMule

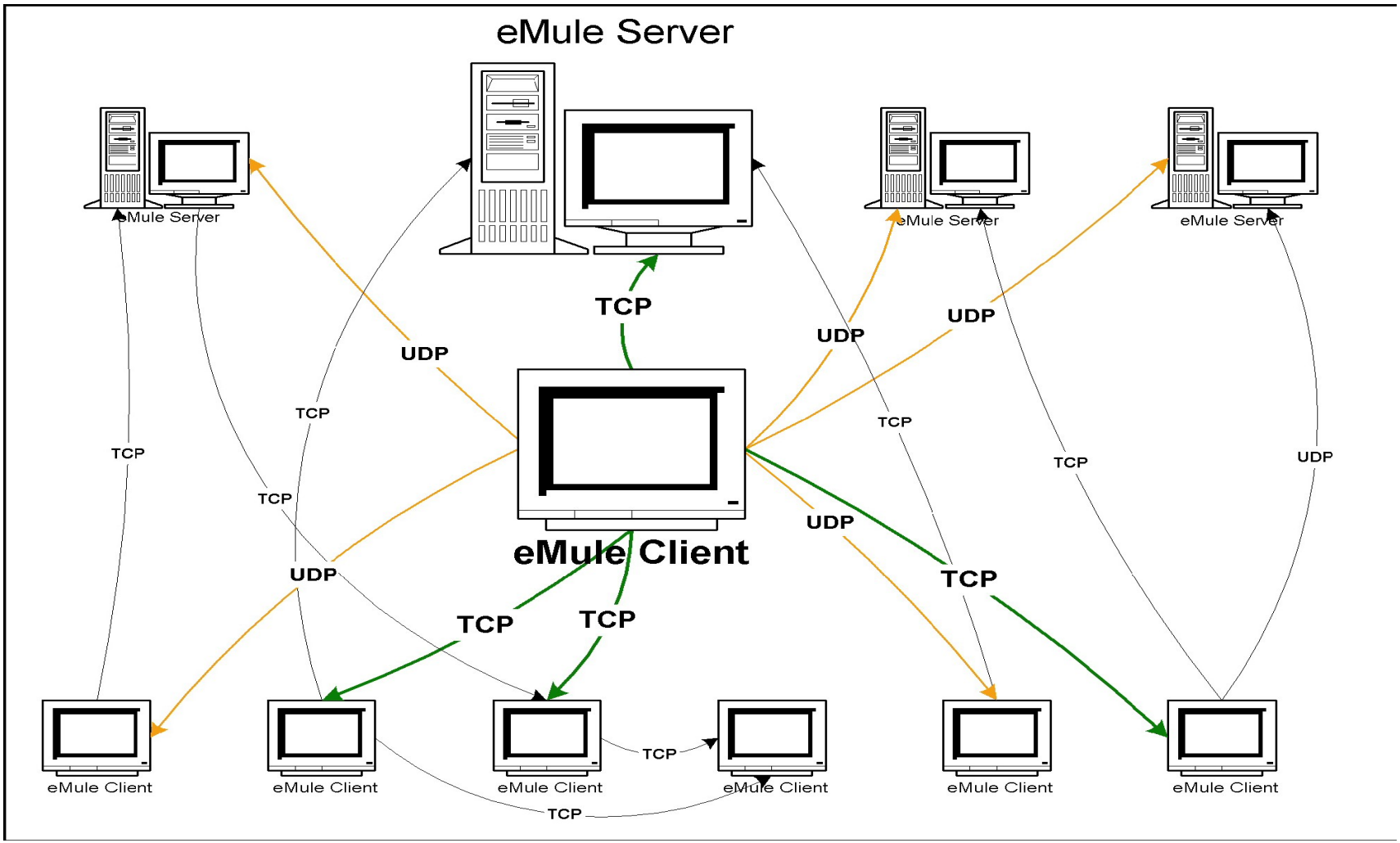
- possiede una lista di servers eMule precaricata
- si connette ad un unico server eMule mediante una connessione TCP. Il server può essere modificato dinamicamente con l'intervento dell'utente
- Apre centinaia di connessioni TCP con altri client eMule per effettuare il la ricerca dei files
- può scaricare frammenti diversi dello stesso file da clients eMule diversi
- può effettuare l'upload di frammenti di un file F anche se il download di F non è ancora stato completato
- invia pacchetti UDP ai servers presenti nella sua lista per verificare la loro presenza sulla rete, per migliorare la ricerca di files,….
- invia pacchetti UDP agli altri clients per controllare il proprio stato nelle code degli altri clients

eMule: IL SERVER

Ogni server eMule

- Possiede un database in cui memorizza i descrittori dei files messi in condivisione dagli utenti ad esso connessi
- Invia ad ogni client eMule, al momento della sua connessione, informazioni circa il numero di utenti connessi ad esso e sul numero di files condivisi
- Non memorizza i files condivisi
- Non interagisce con gli altri server eMule
- Può essere utilizzato come 'punto di appoggio' per utenti a monte di NATs/firewalls

LA RETE ED2k:ARCHITETTURA



IDENTIFICAZIONE DELLE ENTITA' NELLA RETE eMule

USER ID (User Hash)

- stringa di 128 bits (16 bytes) ottenuta concatenando numeri random
- utilizzato per implementare il sistema dei crediti
- assegnato la prima volta che eMule viene lanciato, non viene modificato nelle sessioni successive

CLIENT ID

- Stringa di 32 bits (4 bytes)
- Valida solamente per la durata di una sessione TCP
- Può essere assegnato un ID ALTO oppure un ID BASSO a seconda del tipo di connessione tra l'utente e la rete

FILE HASH

- Stringa hash di 128 bits calcolata mediante l'uso di MD4
- Assegnata dal sistema non appena il file viene messo in condivisione
- Identifica univocamente il file all'interno della rete
- Può essere individuato nella finestra file condivisi, colonna FILE ID

IL PROTOCOLLO CLIENT SERVER

- Ogni client si connette al massimo ad un server per volta
- L'utente può scegliere il server da una lista oppure lasciare la scelta automaticamente al sistema. Successivamente, il server può essere cambiato solo su intervento dell'utente
- Si stabiliscono due connessioni TCP: una dal client al server e viceversa
- Nel momento in cui la connessione viene stabilita il server
  - identifica il client
  - decide se assegnare al client un ID BASSO oppure un ID ALTO, valido per l'intera sessione e determinato in base alla raggiungibilità del client dall'esterno
  - può decidere di rifiutare la connessione con il client (ad esempio quando il numero di connessioni gestite dal server supera una soglia predefinita)

ASSEGNAZIONE DEL CLIENT ID

- Il Server assegna ad un client C un ID basso se C non può accettare connessioni in ingresso (ad esempio si trova a monte di un NAT o di un firewall)
- ID ALTO il client C consente agli altri client dalla rete eMule di connettersi liberamente alla porta TCP eMule 4662
- ID ALTO: viene calcolato a partire dall'indirizzo IP dell'host
  - Se l'indirizzo IP del client è X.Y.Z.W
    - ID = X+ 256*Y + 256*256*Z +256*256*256*W
- L'ID (alto o basso) viene visualizzato al momento della connessione col server
  - **06/05/2007 0.15.35: WARNING DonkeyServer No2 (62.241.53.16:4242) – Your**
  - **20603 port is not reachable. Please review your network config.**
  - **06/05/2007 0.15.35: Connessione stabilita con: DonkeyServer No2**
  - **06/05/2007 0.15.35: Caricamento fonti dal nuovo server in corso per tutti i file con**
  - **meno di 400 fonti**
  - **06/05/2007 0.15.35: Il nuovo ID è 4241605**
  - **06/05/2007 0.15.35: Ricevuti 1 nuovi servers**

SVANTAGGI DI UN ID BASSO

- Un client con ID basso in generale non possiede un indirizzo pubblico a cui gli altri client si possono collegare
- La comunicazione richiede l'intervento del server eMule ed aumenta il carico computazionale sul server.
- Per questa ragione un server accetta poche connessioni da ID bassi e quindi è più facile che la connessione venga rifiutata da un server
- La connessione tra due utenti con ID basso non può avvenire (era possibile nelle versioni precedenti di eMule utilizzando il server come appoggio, poi questa funzionalità è stata eliminata per non sovraccaricare il server)
- un utente con ID basso può effettuare upload di file solo con utenti collegati allo stesso server.

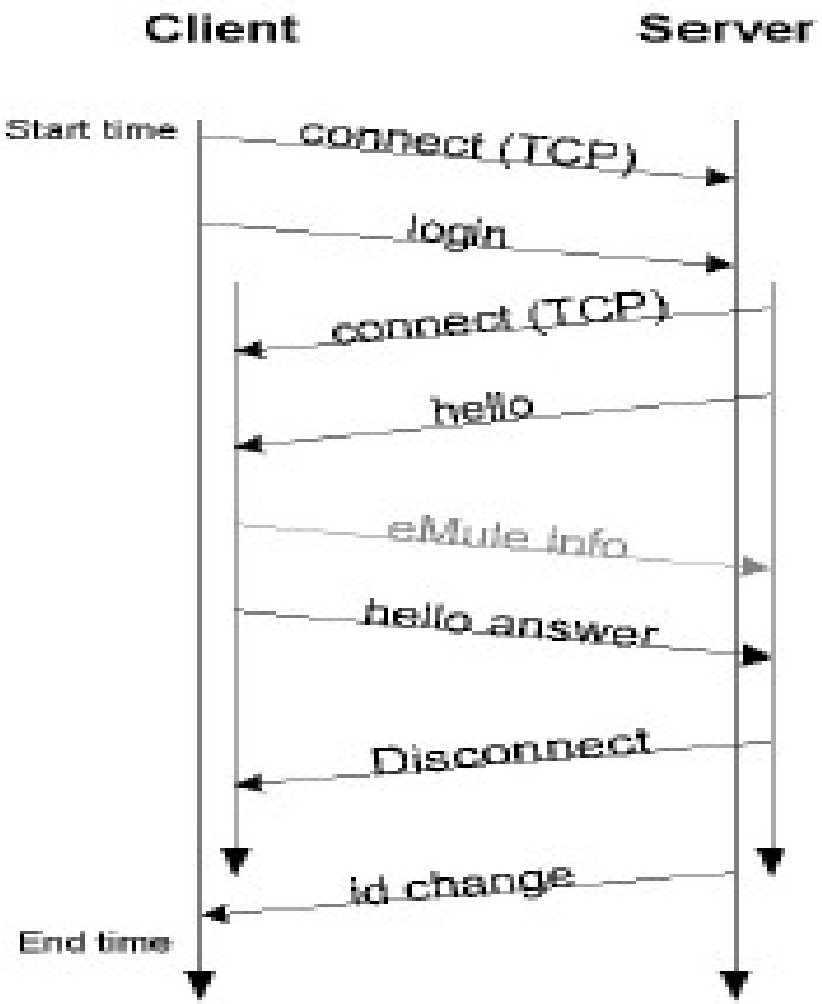
COME FARE PER OTTENERE UN ID ALTO

Aprire le porte sul router/NAT

- Configurare il router/NAT in modo che tutte le comunicazioni provenienti dall'esterno e dirette alle porte TCP 4662 ed UDP 4672

  (o altre porte, se modificate dall'utente) siano instradate verso l'IP corrispondente al computer su cui è in esecuzione eMule
- Se esistono più clients collegati al router mediante una rete locale, occorre associare alle diverse istanze di eMule porte diverse e "aprire" tutte queste porte sul router
- E' inoltre necessario 'spegnere' il DHCP, perchè questo assegnerebbe un indirizzo IP diverso ogni volta che il sistema viene reinizializzato
- L'apertura delle porte spesso può essere effettuata solo dall'amministratore della rete

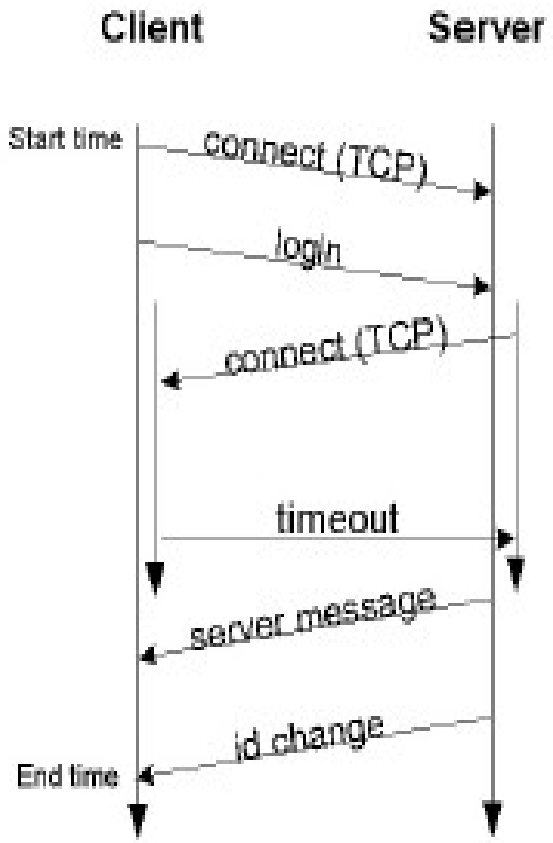
ASSEGNAZIONE CLIENT ID ALTO



ID-CHANGE =
Assegnazione dell'id al client

La connessione aperta dal server verso il client è utilizzata solo per verificare la raggiungibilità del client

ASSEGNAZIONE CLIENT ID BASSO



- Il server non riesce a connettersi al client, Perchè la richiesta di connessione viene bloccata dal NAT

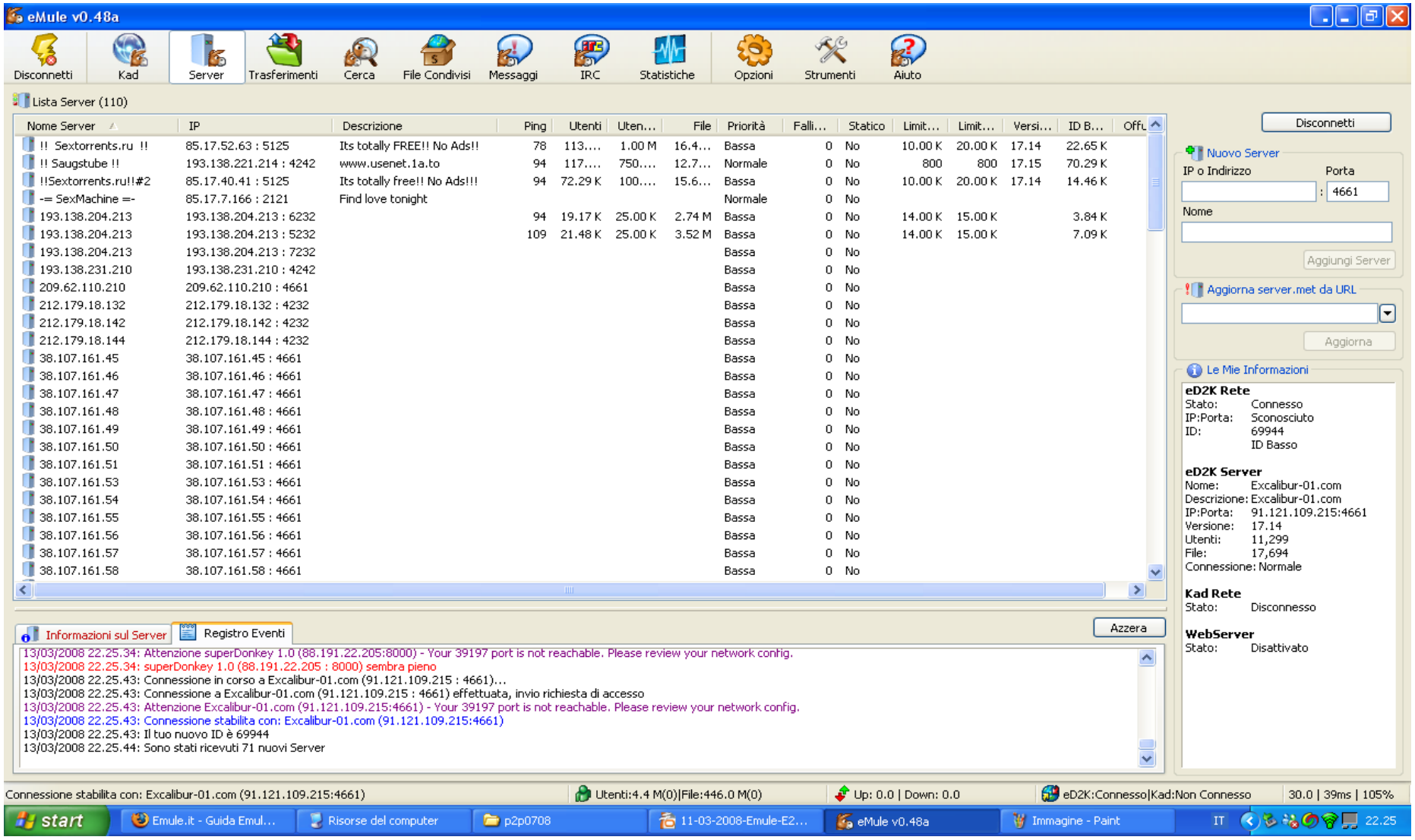
- Il server invia sulla connessione aperta dal client un messaggio del tipo:

  Warning: You have a low ID. Please review your network configuration and/or your settings

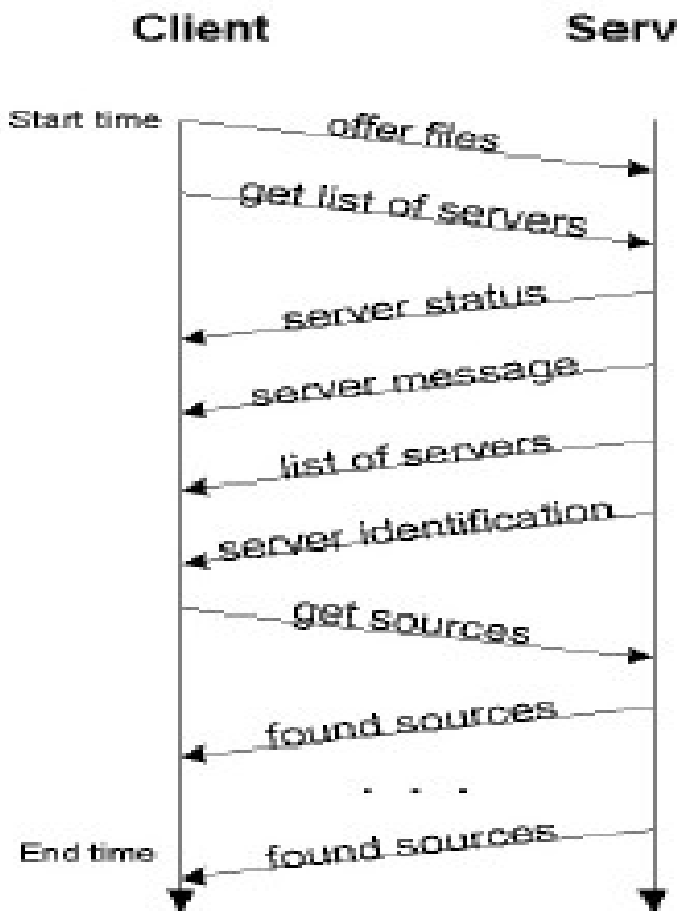
- la fase di connessione si conclude con un messaggio di ID CHANGE che assegna al client un ID basso

Il server può anche decidere di rifiutare la connessione, perchè troppo carico

CONNESSIONE CON IL SERVER



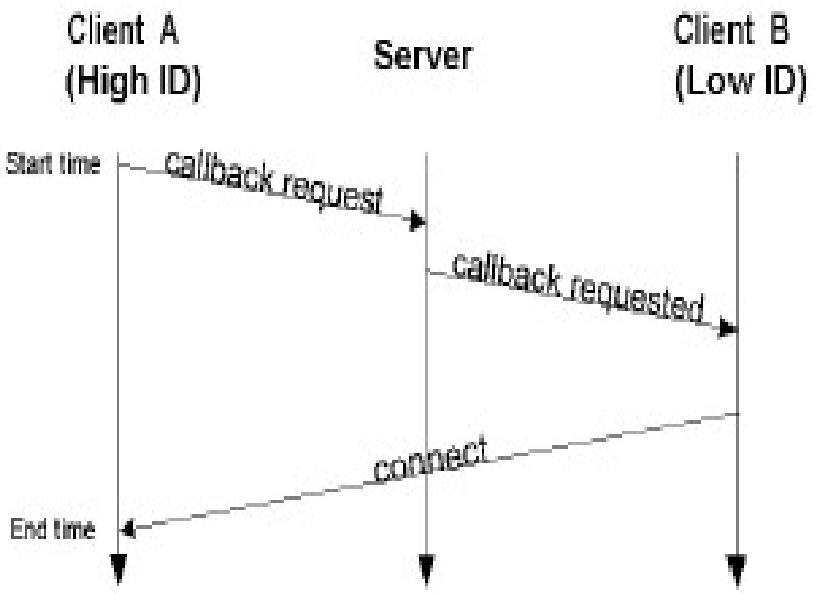
CLIENT-SERVER START UP



- Il client invia la lista dei files L che intende condividere, se L esista già al momento della connessione

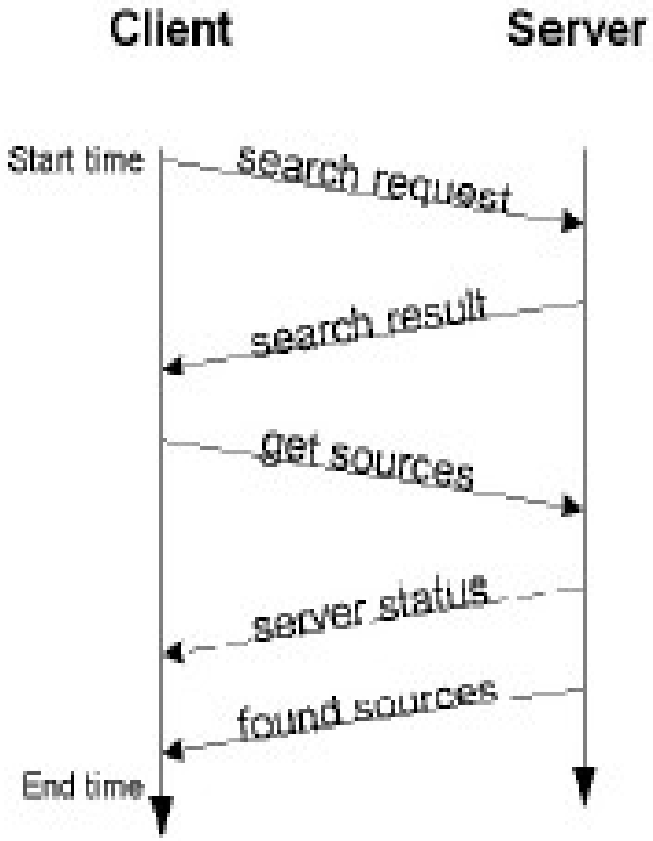
- Il server invia
  - una lista di servers conosciuti
  - nuove fonti per tutti i files nella lista di download del client per cui esso possiede un numero limitato di fonti (valore soglia prestabilito)

CONNECTION REVERSAL MEDIANTE CALLBACKS



- Consentono ad un utente *A con ID alto* di connettersi a *B con ID basso*
-
- A richiede un file memorizzato su B che ha un ID basso
-
- A chiede al server una 'callback' ed il server, che ha già aperto una connessione TCP con B, invia la richiesta di callback a B. B apre una connessione con A
-
- A e B devono essere connessi allo stesso server

INTERAZIONE CLIENT-SERVER: RICERCA DI FILES



- Il client invia una query che può contenere operatori booleani AND, OR, NOT

- il server risponde con una lista di files che soddisfano la query

- il client sceglie uno o più files da scaricare

- il server comunica una lista di fonti per i files prescelti (ogni fonte può possedere tutto il file o una parte di esso)

- status message contiene informazioni sul numero di utenti e di files gestiti dal server

INTERAZIONI UDP TRA CLIENT E SERVERS

La comunicazione UDP tra client e server viene utilizzata per:

Keep alive:

- il client verifica periodicamente lo stato dei servers nella sua lista,
- la richiesta del client include un numero random su cui viene fatto l'eco da parte del server
- ogni volta che il client invia un keep alive al server, incrementa un contatore *C*
- ogni volta che il client riceve un messaggio di qualsiasi tipo dal server, *C* viene decrementato
- se *C* raggiunge un valore soglia, il server viene considerato offline e viene escluso dalla lista dei servers

Aumentare il numero di match per una query

Aumentare il numero di fonti

- se il numero di fonti per un certo file ricevute dal server di riferimento risulta sotto una certa soglia, il client invia pacchetti UDP agli altri servers nella sua server list per ottenere ulteriori fonti

IL SISTEMA DEI CREDITI

- Sistema di crediti: introdotto per incentivare la condivisione dei files da parte degli utenti
- Quando un client *C1* effettua l'upload di un file verso un client *C2*, *C2* 'ricompensa' *C1* mediante l'attribuzione di un credito
- I crediti non sono globali,
- Un credito è un valore intero nell'intervallo 1-10 che viene assegnato ad ogni coppia ordinata (client, client)
- Un client memorizza in un proprio file i crediti degli altri client da cui ha scaricato in precedenza qualche file
- I crediti di proprietà di un client *C* non vengono quindi memorizzati da *C*, ma dai clients che gli hanno concesso il credito, perchè hanno scaricato qualche file da *C*
- Non è quindi possibile visualizzare i propri crediti

IL SISTEMA DEI CREDITI

- Viene introdotto un meccanismo di identificazione sicura dei clients per prevenire la richiesta illecita di crediti da parte di clients che non li posseggono
- eMule tiene traccia, per ogni utente, di tutti i Megabyte forniti, per un periodo di 5 mesi .
- I crediti sono mantenuti in un file nella cartella di configurazione di e-Mule
- All'avvio eMule cancella i crediti scaduti

CALCOLO DEL VALORE DEI CREDITI

- Il valore dei crediti varia da 1 a 10 e viene calcolato mediante 2 formule
- Analizziamo il punteggio che un peer P_1 assegna al peer P_2 per influenzare la posizione di P_2 nella sua coda di upload
- P_1 calcola i seguenti valori
  - TotaleMBytesRicevuti(da P_2) * 2 / TotaleMBytesInviati (a P_2)
  - SQRT (TotaleMBytesRicevuti(da P_2) +2)

  e poi sceglie il valore più basso tra i due
- Condizioni particolari
  - Se TotaleBytesRicevuti < 1MB il valore del credito è 1
  - Se TotaleBytesInviati = 0, il valore del credito è 10

CALCOLO DEL VALORE DEI CREDITI

Esempio: analizziamo il punteggio che il mio client eMule attribuisce ad un client C.

Se ho ricevuto in passato 10MB da C e ne ho inviati 1:

FORMULA 1

- (Megabytes ricevuti da C * 2) / Mega bytes inviati a C

  (10 * 2) / 1 = 20 (cioè 10 dato che il massimo è 10)

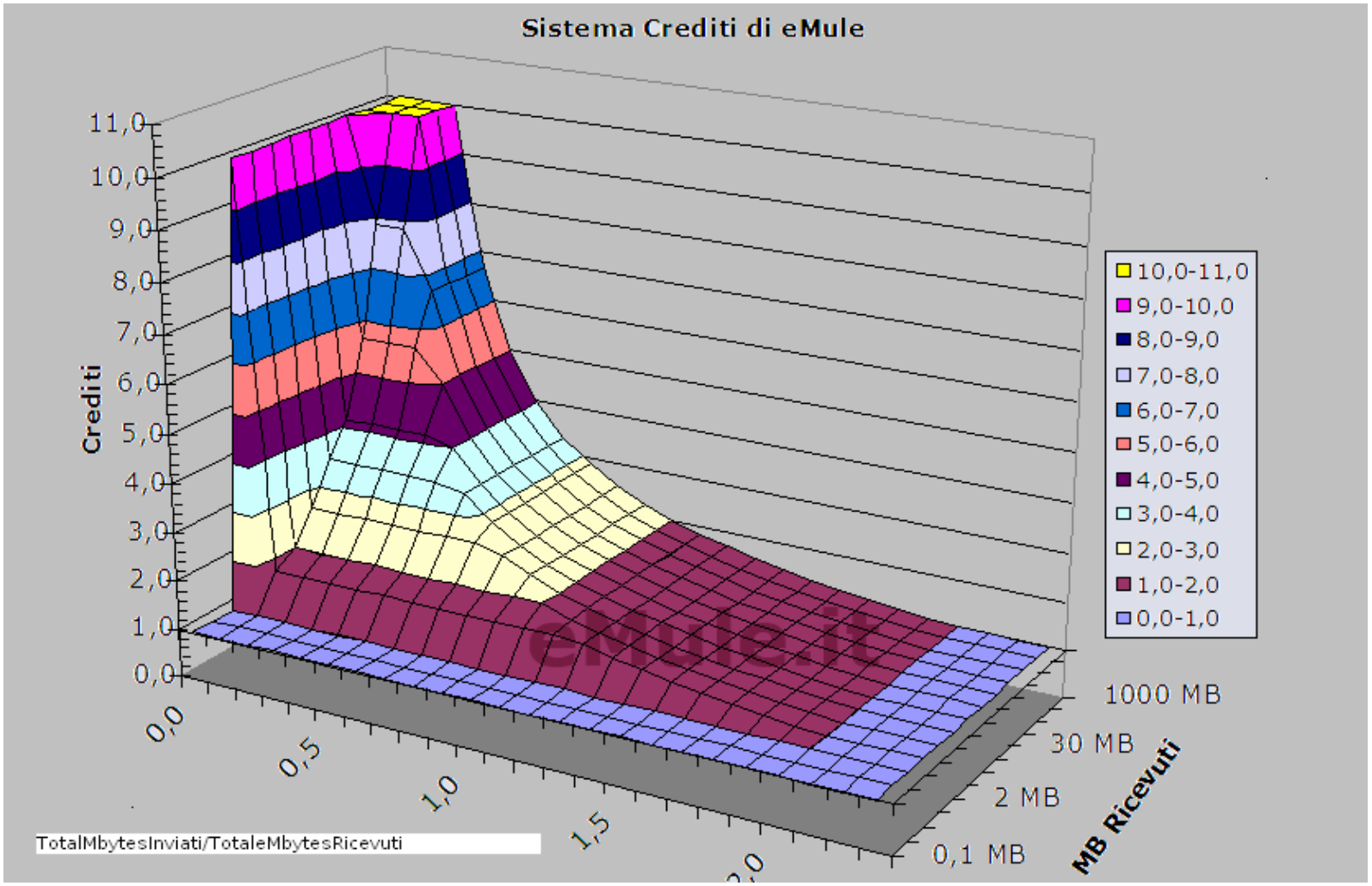
FORMULA2

- crediti = SQRT (Megabyte ricevuti + 2)

  SQRT (10 + 2) = circa 3.5

infine eMule sceglie il valore più basso fra i due calcolati quindi 3.5.

CALCOLO DEL VALORE DEI CREDITI



ANALISI DEL GRAFICO

Sia P è l'utente che sta calcolando il credito da attribuire ad U.

- asse X: rapporto upload/download
  - L'asse X riporta il rapporto tra quanto P ha inviato in passato ad U e quanto ha ricevuto da U
- asse Y: Mbytes che P ha ricevuto da U in passato (servono per la seconda formula)
- asse Z: crediti che P attribuisce ad U
- Valori < 1 sull'asse delle X implicano che P è stato 'avaro' con U (ha inviato meno di quanto ha ricevuto). Il credito attribuito ad U risulta maggiore
- Valore massimo attribuito ai crediti = 10
  - Il valore 10 viene attribuito per default ad U se P non gli ha mai inviato niente in precedenza

ANALISI DEL GRAFICO

Se il rapporto bytes inviati/bytes ricevuti è:

- ≥ 2 P ha dato almeno il doppio di ciò che ha ricevuto, quindi ad U viene assegnato un crediti =1, cerdito minimo

- compreso fra 1 e 2 P è stato abbastanza generoso con U, i crediti di U presso P variano da 2 a 1, cioè un valore medio

- < 1 P è stato 'avaro' con U, i crediti variano da 2 a 10.
  - Il valore dei crediti cresce in maniera proporzionale al numero di bytes ricevuti in passato da U
  - Il valore 10 è raggiunto solo se U ha fornito almeno 98MB.

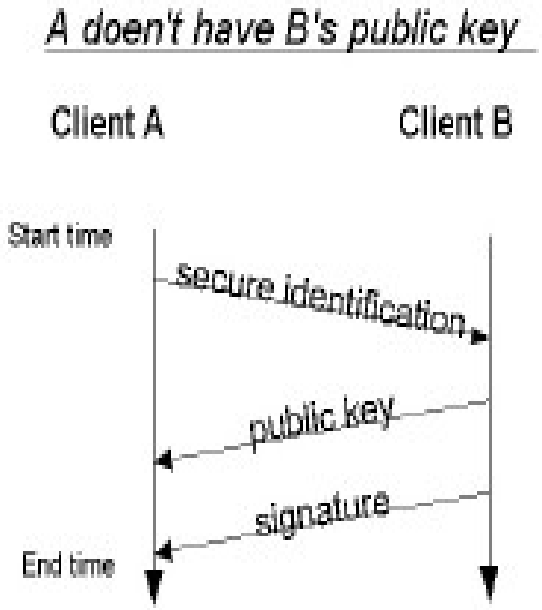
SECURE CLIENT IDENTIFICATION

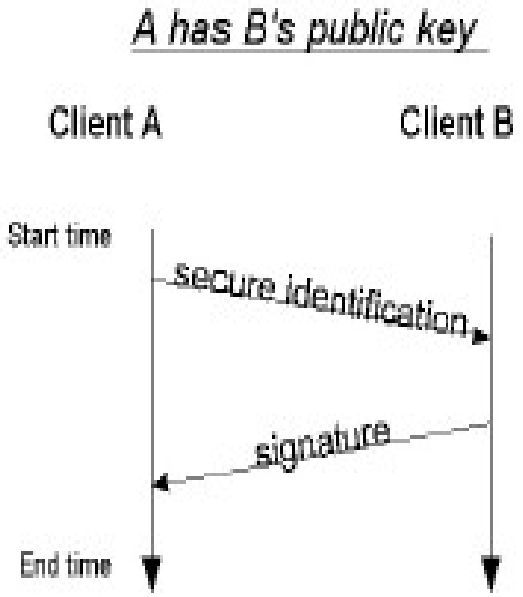
- E' necessario utilizzare un sistema di identificazione sicura dei clients per evitare che un client si impossessi dei crediti destinati ad un altro client

- Basato su un sistema a crittografia asimmetrica (RSA)

- Supponiamo che un client B si deva autenticare presso il client A
  - B invia la sua identità e la firma generando la firma mediante la sua chiave privata,
  - A reperisce la chiave pubblica di B
  - A utilizza la chiave pubblica di B per decodificare la firma di B e verificare l'identità di B
  - eMule crea in automatico alla prima connessione la coppia chiave pubblica-chiave privata, che viene memorizzata nella cartella config di eMule.

SECURE CLIENT IDENTIFICATION





COMUNICAZIONE CLIENT/CLIENT

- La maggior parte dei messaggi viene spedita su connessioni TCP
- Viene stabilita una diversa connessione TCP per ogni coppia [file, client1, client2]
- Connection startup: handshake simmetrico in cui i due clients si identificano mediante Secure User Identification
- Dopo l'handshake, il client C indica al partner P quale file intende scaricare
- P inserisce C nella sua coda di upload
- Nella versione ufficiale di eMule esiste una coda di upload unica per tutti i clients

GESTIONE DELLE CODE

- mantiene una singola coda in cui inserisce tutte le richieste di upload
- coda di upload = coda a priorità
- in ogni istante serve un numero limitato di download, in modo da non esaurire tutta la banda a sua disposizione
- quando un client A richiede il download di un file da B, la sua richiesta

  viene servita immediatamente se la coda di upload di B è vuota ed il numero di upload attivi in B non supera una certa soglia
- altrimenti la richiesta viene inserita nella coda di upload di B

  B invia ad A un messaggio indicando la posizione di A nella propria coda

INTERAZIONE TRA CLIENTS: GESTIONE DELLE CODE

- Priorità = (rating * waiting time) / 100
- La priorità di un client nella coda di upload dipende da
  - il tempo di attesa in coda (waiting time)
  - un rating che dipende
    - dal credito acquisito dal client downloader CD presso il client uploader CU (1-10)
    - dalla priorità del file che si sta scaricando (0.2-1.8)
    - in genere, rating = credito acquisito * priorità del file
- Il download inizia quando il client raggiunge la prima posizione della coda
- Il downlaod continua finchè
  - Il client interrompe il downlaod
  - Preemption: un client con una priorità più alta richiede il downlaod

FILE CONDIVISI: PRIORITA'

- Per ogni file che si sta condividendo è possibile impostare una priorità (release, alta, normale, bassa, molto bassa, auto)
- La priorità indica quanto l'utente vuole 'favorire' l'upload di un file
- La priorità può essere settata per ogni file che appare nella finestra dei file condivisi cliccando con il tasto destro sul nome del file
- Per default eMule imposta la priorità auto (automatica)
- Priorità Release: utilizzata da chi mette in condivisione un file per la prima volta (releaser)
- Priorità automatica: emule cambia automaticamente la priorità a seconda di quanto un file è popolare ed in questo modo favorisce:
  - lo scaricamento dei file più rari che sono più difficili da ottenere,
  - la diffusione generale dei file nella rete eMule
- Emule indica per ogni file condiviso il numero di fonti: è possibile aumentare la priorità di file molto rari

INTERAZIONE TRA CLIENTS: GESTIONE DELLE CODE

- Nel client eMule, nella finestra downloads è possibile individuare, accanto al nome del file in download il suo QR (Queue Rank)
- QR= 100, esistono 100 utenti in coda prima che devono essere serviti prima che la mia richiesta venga presa in considerazione
- QR da un'approssimazione del tempo di attesa, infatti è possibile che altri utenti mi superino sulla coda perchè possiedono un numero maggiore di crediti

INTERAZIONE TRA CLIENTS: SCAMBIO FONTI

- Una connessione diversa per ogni coppia [file, client]
- A invia l'identificatore del file che intende scaricare
- B risponde con informazioni relative al file richiesto (status)
- Inoltre B passa ad A le fonti di A di sua conoscenza.

INTERAZIONE TRA CLIENTS: SCAMBIO DELLE FONTI

- Ogni volta che richiedo il download di un file, eMule ricerca le fonti per quel file tramite i servers contenuti nella lista del client
- La ricerca avviene subito sul server di riferimento e, ad intervalli più lunghi, sugli altri servers della lista (mediante comunicazione UDP)
- Inoltre, quando si contatta una fonte per il download, il client richiede alla fonte contattata la lista delle fonti in suo possesso per il medesimo file
- Ciò avviene sia se la fonte contattata è completa sia se solo alcune parti di quel file sono disponibili su quella fonte..
- Conoscenza incrementale delle fonti: comportamento gossip-based

VANTAGGI

- Ricerca alternativa di fonti: percentuale di fonti contattate su quelle disponibili nella rete eMule molto prossima al 100%.
- Maggiore velocità di contatto, non dovendo passare per i server.
- Minor carico di lavoro dei server, con evidente miglioramento delle prestazioni del sistema.

TRASFERIMENTO DI FILES TRA CLIENTS

- Un client può
  - scaricare solo una parte del file da un altro client
  - scaricare parti diverse dello stesso file da utenti diversi
- Le richieste di download si riferiscono a parti di files
- Tutti i blocchi scaricati da uno stesso client appartengono alla stessa parte
- Quando il downloader D richiede ad un uploader U il download di un file F, U può possedere solo alcune parti del file, perchè ne sta effettuando il download
- U invia a D un messaggio in cui indica se possiede tutto il file ed, in caso negativo quali parti possiede
- D può richiedere a U il download di alcune parti del file (massimo 3 parti per ogni messaggio)

TRASFERIMENTO DI FILES TRA CLIENTS

- Il client che effettua il download determina
  - quali parti scaricare del file selezionato
  - l'ordine con cui le parti vengono scaricate
  - da quale client vengono scaricate

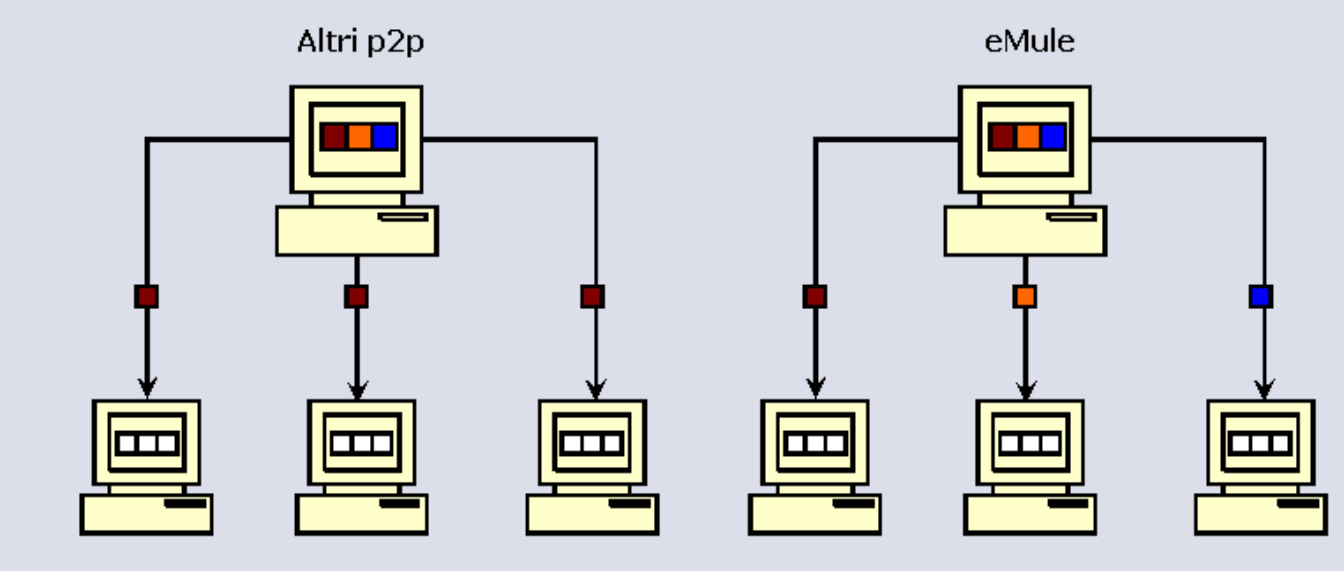
Selezione delle parti da scaricare

- Distribuire le richieste di parti tra le sorgenti conosciute
- Scaricare per prime le parti più rare
- Prima la prima e l'ultima parte

Part rating: viene attribuito un punteggio diverso ad ogni parte che dipende dalla

- Disponibiltà delle parti
- Parti importanti (prima e ultima parte)
- Meccanismi più sofisticati introdotti da Bittorrent (prossime lezioni)

eMule: UPLOAD DEI FILES

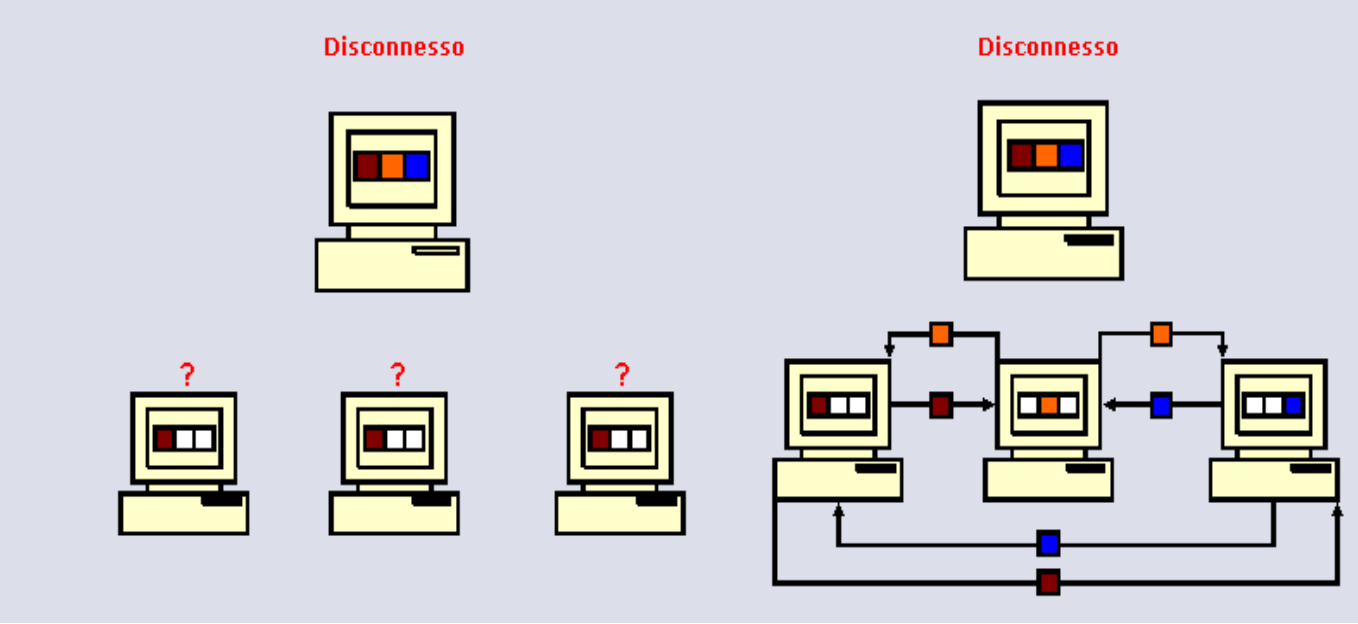


- strategia di uploading:

se un file viene richiesto da più di un downloader, eMule cerca di inviare a ciascun downloader una parte diversa del file

- in questo modo si tende a distribuire più rapidamente il file nella rete e-Mule

eMule: UPLOAD DEI FILES



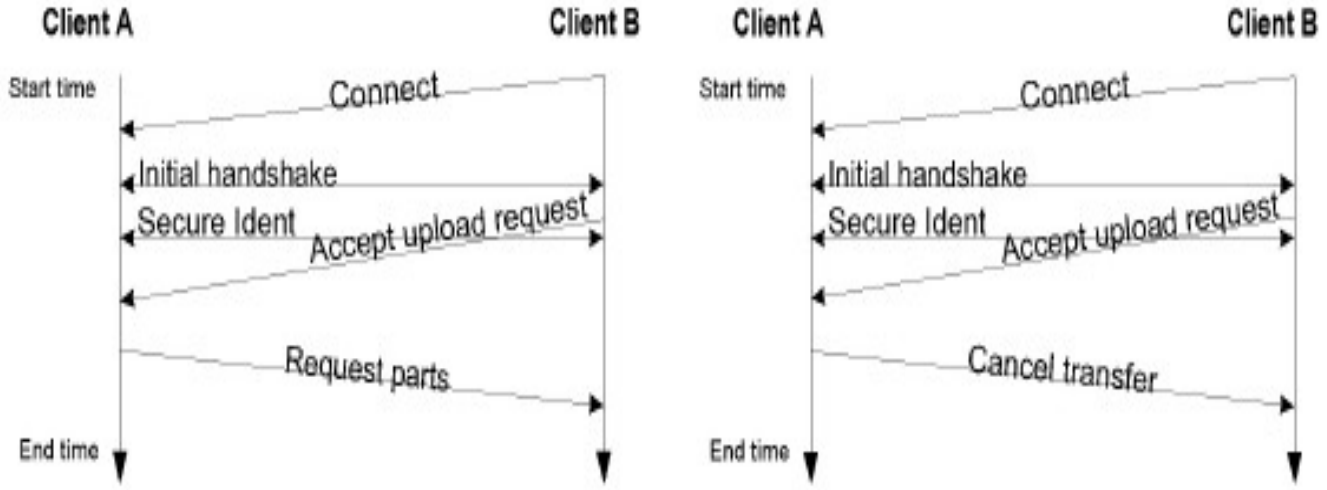
Supponiamo che l'uploader U si disconnetta dopo aver inviato una parte del file ad ogni client eMule

- se U ha inviato la stessa parte P del file F a tutti i clients, tutti si possiedono solo P e devono attendere la connessione successiva di U per poter scaricare F (figura di sinistra)
- se U ha distribuito parti diverse a client diversi, il file è presente nella rete ed i clients possono scambiarsi le parti in modo da ricostruire il file

TRASFERIMENTO DI FILES TRA CLIENTS

- Quando inizio il download di un file, vengono contattate più fonti e la richiesta viene inserita nelle code di upload di tutte le fonti
- Quando il client *C* ha completato il download dell'intero file, non lo notifica agli altri clients
- Quando un client invia il messaggio di upload a *C*, la proposta di uplaod viene rifiutata da *C*

TRASFERIMENTO DI FILES TRA CLIENTS



- Quando A raggiunge la prima posizione nella coda di upload di B

B apre una connessione con A

- B chiede una connessione ad A

- A può accettare la connessione , oppure rifiutarla, se il file è già stato scaricato da altre fonti

IDENTIFICAZIONE DEI FILES NELLA RETE E-MULE

- per identificare un file in una rete P2P, non è sufficiente usare il nome del file,
  - file diversi con lo stesso nome
  - file con medesimo contenuto con nomi diversi.

- Link e2k (eDonkey 2000) : identifica in modo univoco un file all'interno della rete di client eMule.

- Forma generale di un link ed2k:

  e2k//|file|<nome file>|<dimensione file>|<file hash>|

Esempio: ed2k://|file|ubuntu-8.10-dvd-amd64.iso|4632141824|
133C2012860F65663FF0962D6FC52231|/

  - ubuntu.....nome del file
  - 46.... dimensione del file
  - 133C2...hash del file

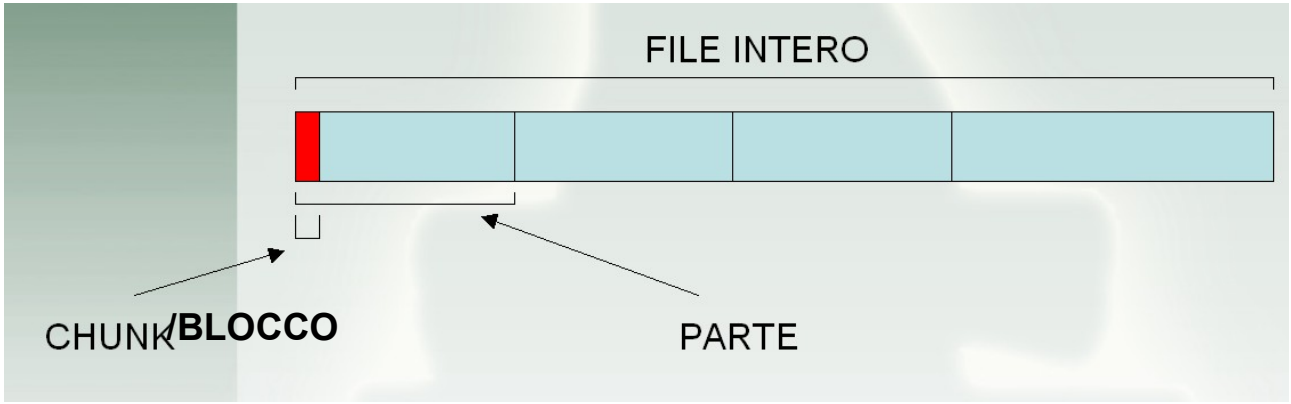
LINK E2K

- identificano in modo univoco i file condivisi della rete eDonkey
  - non indicano in modo esplicito un indirizzo dal quale poter scaricare il file
  - contengono tutte le informazioni necessarie per poter eseguire una ricerca del file

- possono essere
  - individuati cliccando sul tasto destro per i file in download
  - creati con appositi programmi (linkcreator)
  - integrati in una pagina web
    - la maggior parte dei browser invoca automaticamente e-mule non appena viene selezionato un link e2K, mettendo automaticamente in download il file corrispondente

- Diversi tipi di link e2k

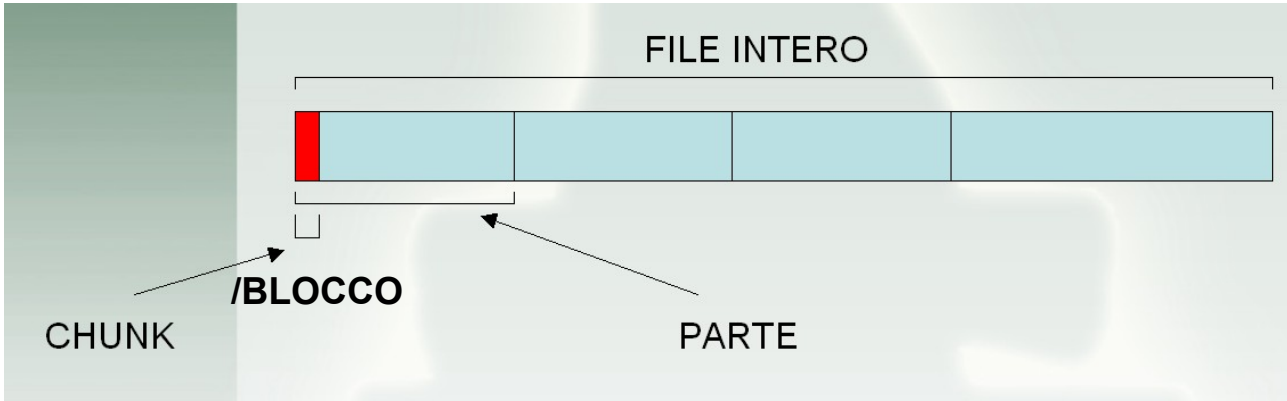
TRASFERIMENTO DI DATI TRA I CLIENTS



- ogni file presente nella rete eMule viene decomposto in parti e quindi ogni parte viene frammentata
  - parti da 9.28 MB
  - ogni parte è a sua volta divisa in chunks(blocchi) da 180K bytes (53 blocchi per parte)
- per ogni parte viene calcolato un hash mediante MD4
- HashSet: contiene gli hash di ogni parte del file
- Link e2k con HashSet

  ed2k://|file|<nome file>|<dimensione file>|<file hash>|p=<hash set>|

TRASFERIMENTO DI DATI TRA I CLIENTS



Esempio di link e2k riferito ad un file composto da una sola parte

- ed2k://|file|ubuntu-5.10-install-i386.iso|lungh.file|901E6AA2A6ACD.....

Esempio di link e2k riferito ad un file composto da più parti

- ed2k://|file|ubuntu-5.10-install-i386.iso|lungh.file|901E6AA2A6ACD.....|
  p=264E6F6B.....:17B9A4D1DCE0E4C.....|/

riporta l'hash dell'intero file e quello di ogni sua parte

INTELLIGENT CORRUPTION HANDLING

- Idea di base della ICH: scaricare di nuovo tutti i chunks della parte,fino a che la parte non risulta riparata
- Il client scarica l'hashset del file prima di una qualsiasi sua parte
- Ogni volta che un client scarica una parte del file, calcola l'hash H di quella parte e confronta H con il valore corrispondente contenuto nell'hash set
- Se i valori coincidono, la parte è integra e può essere messa in condivisione
- Se i valori non coincidono, si ripetere il download della parte, un chunk alla volta
- Per ogni chunk/blocco scaricato viene ricalcolato l'hash su tutta la parte, se il valore calcolato coincide con quello nell'hash set, la parte viene considerata 'riparata' e si evita di trasferire il resto dei chunks di quella parte
- Questo consente di evitare di ripetere in media il 50% del download dei chunks del file

ADVANCED INTELLIGENT CORRUPTION HANDLING

- Problema dell'ICH:
  - quando l'errore è localizzato alla fine di una parte del file, ad esempio a 8,9 MB, l'ICH riscarica tutti i chunks delle parte
  - questo problema è dovuto al fatto che la porzione minima su cui viene calcolato l'hash è la parte di file e non il chunk

- AICH= Advanced Intelligent Corruption Handling:
  - introdotto dalla versione 0.44
  - basato sull'uso di Merkle trees

MERKLE TREE: TEORIA

- Hash trees or Merkle trees: struttura dati che riassume in modo sintetico informazioni su una grossa molr di dati (es: un file) ed il cui scopo è verificarne il contenuto

- Introdotto da Ralph Merkle nel 1979

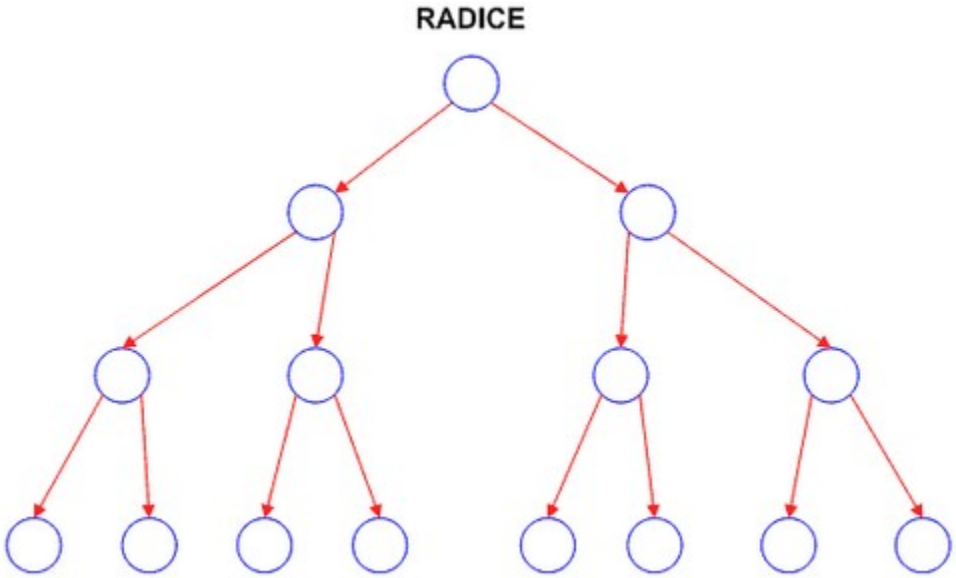
- Caratteristiche:
  - semplicità
  - velocità di calcolo
  - versatilità

- Merkle Tree: albero binario completo costruito a partire da un insieme di 'simboli' iniziali
  - utilizzano una funzione hash H (SHA1, MD5)
  - nodi foglia: H applicata ai simboli iniziali
  - nodi interni: H applicata ai due figli di un nodo

ALBERI BINARI COMPLETI: RICHIAMI

Un albero binario completo di altezza h è caratterizzato dalle seguenti proprietà

- ogni nodo ha esattamente due nodi figli
- le foglie sono tutte allo stesso livello
- nodi foglia: 2^h
- nodi interni: 2^h-1
- nodi totali: $2^{h+1} - 1$

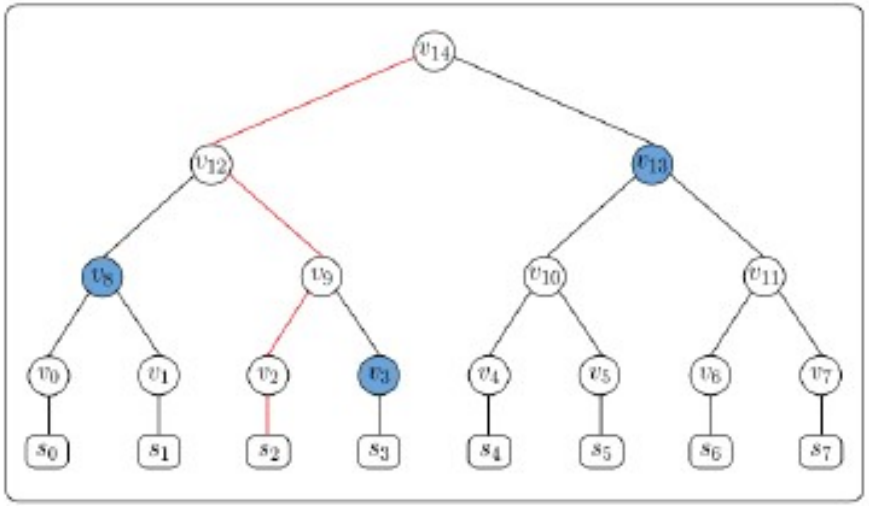


MERKLE TREE: TEORIA

- si consideri un insieme S iniziale di n simboli, $S=\{s_1,....s_n\}$, $n=2^h$, h altezz dell'albero
- i "simboli" possono essere il risultato della frammentazione di un blocco iniziale d dati
- algoritmo di Merkle MHT = <CMT,DMT>
- CMT (Coding Merkle Tree): a partire dall'insieme S dei simboli inziali, costruir l'albero binario completo di altezza h
  - Nodi Foglia = $H(s_i)$
  - Nodi Interni = hash sulla concatenazione dei valori hash dei nodi figli H(l||r)

- Output
  - la radice dell'albero binario
  - un 'testimone' w_i per ogni simbolo s_i
  - w_i = nodi fratelli dei nodi che costituiscono il cammino dalla foglia s_i alla radice

MERKLE TREE: TEORIA

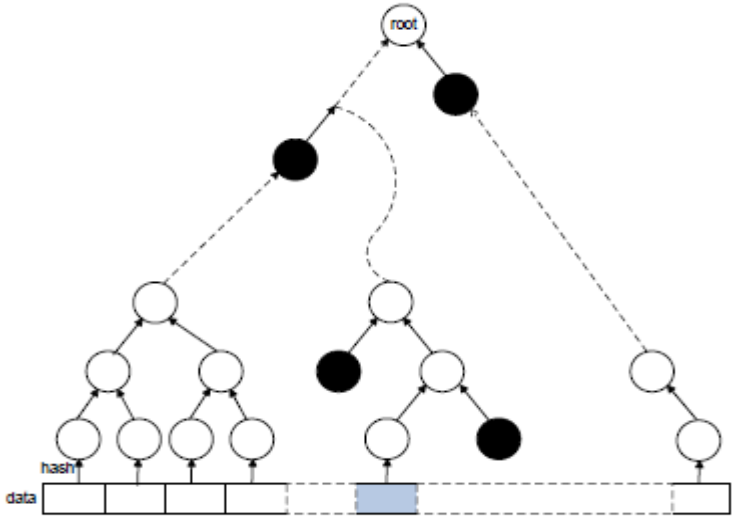


Testimoni	Nodi
$s_0\ w_0 = \{v_1, v_9, v_{13}\}$	$v_0 = h(s_0)\ v_8 = h(v_0 \parallel v_1)$
$s_1\ w_1 = \{v_0, v_9, v_{13}\}$	$v_1 = h(s_1)\ v_9 = h(v_2 \parallel v_3)$
$s_2\ w_2 = \{v_3, v_8, v_{13}\}$	$v_2 = h(s_2)\ v_{10} = h(v_4 \parallel v_5)$
$s_3\ w_3 = \{v_2, v_8, v_{13}\}$	$v_3 = h(s_3)\ v_{11} = h(v_6 \parallel v_7)$
$s_4\ w_4 = \{v_5, v_{11}, v_{12}\}$	$v_4 = h(s_4)\ v_{12} = h(v_8 \parallel v_9)$
$s_5\ w_5 = \{v_4, v_{11}, v_{12}\}$	$v_5 = h(s_5)\ v_{13} = h(v_{10} \parallel v_{11})$
$s_6\ w_6 = \{v_7, v_{10}, v_{12}\}$	$v_6 = h(s_6)\ v_{14} = h(v_{12} \parallel v_{13})$
$s_7\ w_7 = \{v_6, v_{10}, v_{12}\}$	$v_7 = h(s_7)$

MERKLE TREE

- DMT(s_i,w_i,root): Decoding Merkle tree
- Validità di s_i dedotta dal confronto della radice calcolata durante la fase di decodifica in base a w_i e la radice generata in fase di codifica
- Prevede che il simbolo s_i venga autenticato sulla base del testimone w_i e della radice root

MERKLE TREE

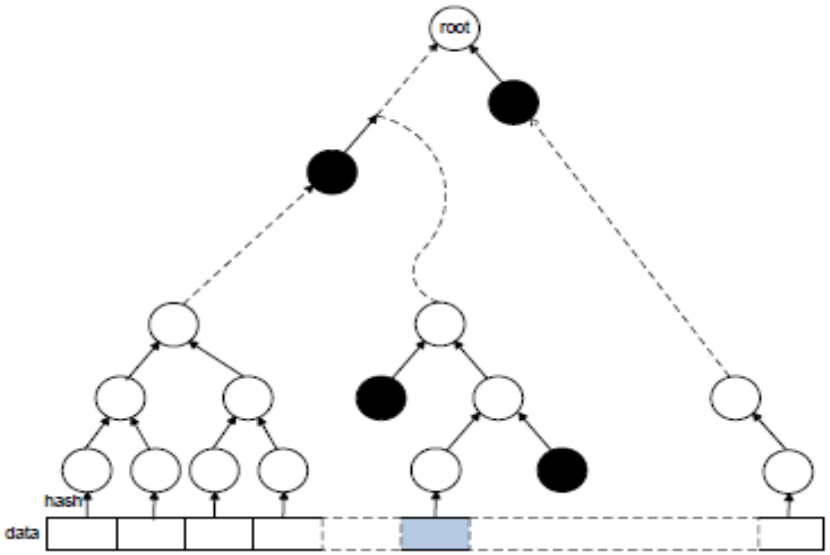


- Merkle Tree albero binario di nodi etichettati, in cui le etichette sono stringhe binarie di lunghezza k
- $\Phi(n)$ etichetta associata al nodo n
- etichetta di un nodo interno dell'albero, $n_{parent,}$ con figli n_{left} ed n_{right}

$$\Phi(n_{parent}) = H(\Phi(n_{left}) \,||\, \Phi(n_{left}))$$

con H funzione hash

MERKLE TREE



Etichette delle foglie sono dipendenti dalla applicazione

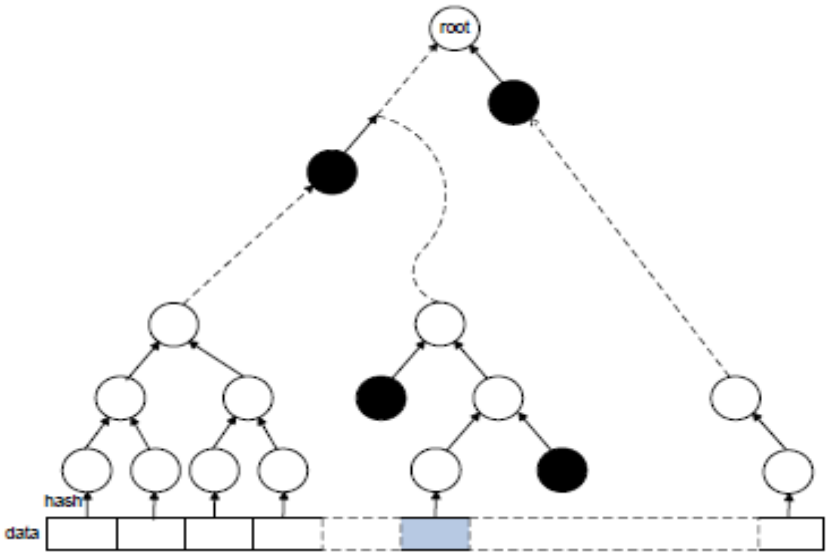
Esempio: hash di una piccola parte di un documento (blocco)

Utilizzati per stabilire l'autenticità delle etichette dei nodi foglia

Viene pubblicato il root hash su un sito considerato sicuro

L'autenticità di una foglia viene stabilita conoscendo il root hash + un alcuni nodi relativi ad un cammino sul Merkle Tree

MERKLE TREE

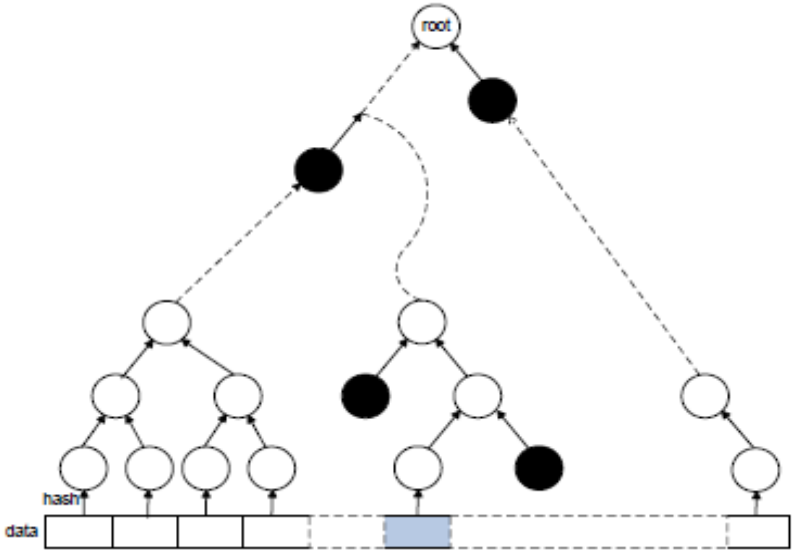


Ad esempio per verificare l'autenticità della foglia evidenziata in grigio, occorre considerare il cammino da x alla radice

Per ogni nodo n su questo cammino si considera il nodo fratello di n nell'albero (i nodi indicati in nero nell'albero)

I nodi neri costituiscono l'authentication path o testimone di x

MERKLE TREE



per verificare l'autenticità di x si calcolano le etichette dei nodi interni, mediante H. Se il valore calcolato per la radice del Merkle tree coincide con il quello reperito mediante un sito sicuro, x viene reputato non corrotto

Se qualcuno avesse corrotto x, sarebbe stato computazionalmente molto pesante individuare un authentication path che restituisca il root hash corretto a partire dal valore di x corrotto

AICH:ADVANCED INTELLIGENT CORRUPTION HANDLING

- utilizza un hash set "costruito" a partire da blocchi di 180KB e raggruppati in una struttura ad albero, un Merkel Tree

- eMule crea l'hash set di ogni file condiviso e lo memorizza in un file

- La dimensioni di questo set di hash può essere notevole
  - 24.000 hash per un file di 4 GB
- Parti dell'albero vengono inviate ai client che stanno scaricando il file

- Se eMule scarica un file e rileva una parte corrotta, cerca un "pacchetto di verifica"
  - contiene gli hash dei blocchi della parte
  - altri valori hash necessari per verificare se il pacchetto stesso è a sua volta integro

AICH:ADVANCED INTELLIGENT CORRUPTION HANDLING

- l'hash set dei blocchi delle parti non può essere ricavato dal link e2K
  - risulterebbe troppo dispendioso inserire in un link e2K un hashset per tutti i blocchi del file
  - il link e2K contiene unicamente il root hash
  - link e2k con root hash

  ed2k://|file|<nome file>|<dimensione file>|<file hash>|h=<Root Hash>|/

- si suppone che il root hash sia sicuro, reperito da un sito affdabile

- l'hash set dei blocchi della parte corrotta viene scaricato da un altro peer, il peer che ha creato il file e possiede l'hash set completo del file

- Occorre verificare che l'hash set non sia a sua volta corrotto
  - Verifica effettuta mediante Merkle trees

AICH:ADVANCED INTELLIGENT CORRUPTION HANDLING

- quando il client che ha rilevato una parte corrotta riceve il pacchetto di verifica
  - verifica la corrispondenza tra pacchetto di recupero ed hash AICH
  - confronta l'hash di ogni blocco con gli hash contenuti nel pacchetto di verifica: conserva tutti i blocchi non corrotti, ad eccezione dell'unico blocco corrotto, che riscarica

- Ad esempio, se un solo byte è corrotto, eMule mantiene tutti i blocchi da 180KB "sani", ad eccezione dell'unico blocco da 180KB contenente il byte corrotto, blocco che viene riscaricato.

- Oppure, nel caso in cui il chunk corrotto si trovi in fondo al file, viene riscaricato solo l'ultimo chunk da 9.1 a 9.28 MB invece di tutti e 9.2 MB.

- Vantaggio: tramite l'A.I.C.H. è possibile recuperare parti di file senza dover essere costretti a riscaricare blocchi non corrotti.

VERIFICA INTEGRITA' DEL PACCHETTO

- il pacchetto di recupero contiene
  - tutti gli hash dei chunks della parte corrotta (53 block hashes)
  - alcuni hash di verifica dell'intero albero hash (numero hash di verifica x tale che $2^{x} \geq$ numero di parti del file)
  - il numero di hash di verifica dipende dal numero di parti del file

- processo di verifica
  - risale l'albero combinando i risultati con gli hash di verifica, fino ad ottenere il root hash
  - confronta il root hash con quello in suo possesso, se il root hash coincide,il pacchetto è integro e può essere utilizzato per il recupero della parte

AICH:ADVANCED INTELLIGENT CORRUPTION HANDLING

- Problema:diffusione di root hash corretti
- Reperiti mediante i link e2k
- Ogni client richiede agli altri client il root hash di un file
  - Se almeno altri 10 client mi inviano lo stesso Root Hash e questo Root Hash è uguale ad almeno il 92% o più dei Root Hash ricevuti,allora quel root hash viene considerato affidabile

COMUNICAZIONE UDP CLIENT/CLIENT

- Un client *C* invia periodicamente pacchetti UDP ai clients da cui sta tentando di scaricare i files
- Il pacchetto UDP viene inviato per conoscere lo stato della richiesta di download effettuata da *C* presso *C1*
- Possibili risposte
  - La posizione di *C* nella coda di upload di *C1*
  - La coda di upload di *C1* è piena
  - *C1* non possiede il file richiesto da *C*

LA RETE KAD

- Nella rete KAD l'indice dei file condivisi non è più memorizzato su un insieme di servers, ma su una rete (overaly) costituita dagli stessi clients eMule

- Scopo principale: eliminare qualsiasi punto di centralizzazione dalla rete
  - evitare eventuali procedimenti legali dovuti alle nuove leggi sui copyright in circolazione in vari Paesi del mondo.
  - i 30 servers più grossi contengono oltre il 90% degli utenti
    - problemi di carico eccessivo

- Rete KAD utilizza una tecnologia basata sull'uso di Distributed Hash Tables
  - Basata sulla DHT Kademlia
  - Replicazione delle entrate dell'indice a causa del peer churn