Lezione n.3
Sistemi P2P Ibridi
Gnutella 0.6- Kazaa

Tutorial Kazaa sulla pagina web del cosro

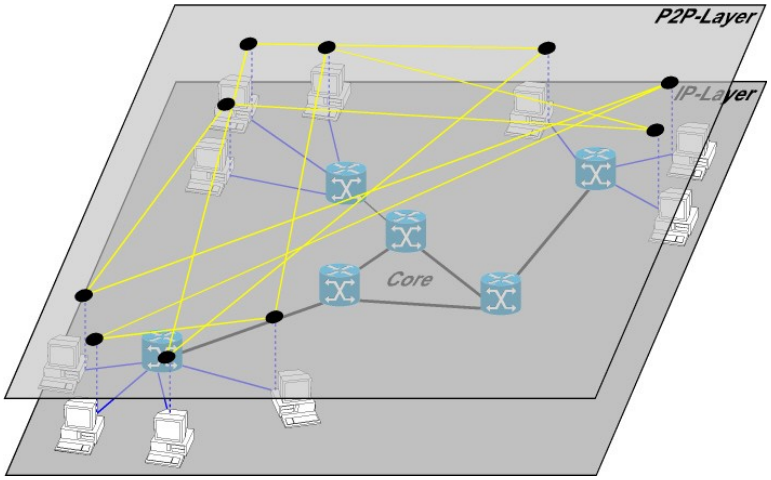
Laura Ricci
9/10/2013

COSA VEDREMO IN QUESTA LEZIONE

- Reti P2P gerarchiche
- Protocollo Gnutella 0.6
- Elezione Dinamica di SuperPeer: Self Organizing Network
- Utilizzo dei Bloom filters
- Kazaa

RIASSUNTO DELLA PRESENTAZIONE

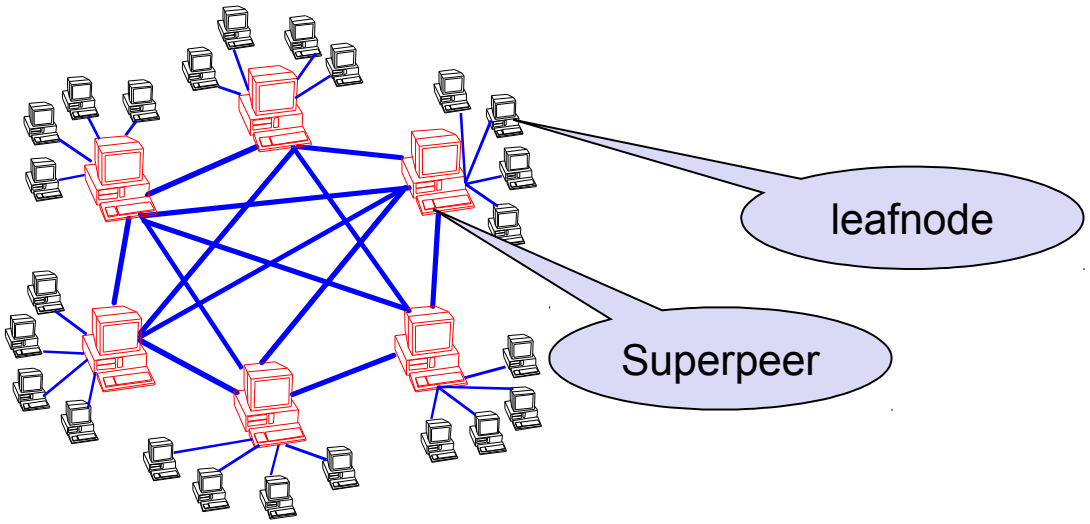
1. Caratteristiche Generali dei Sistemi P2P
2. Reti P2P Centralizzate
   - Caratteristiche Base
   - Protocolli
   - Discussione
1. Reti P2P pure
   - Caratteristiche Base
   - Protocolli
   - Discussione
1. Reti P2P Ibride
   1. Caratteristiche Base
   2. Protocolli
   3. Discussione



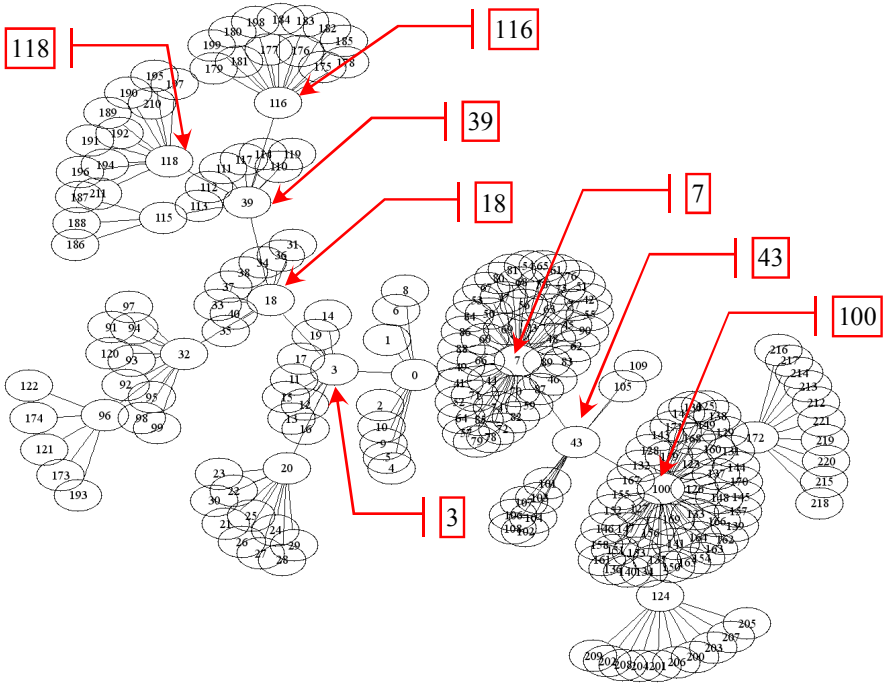
GNUTELLA 0.6: CARATTERISTICHE GENERALI

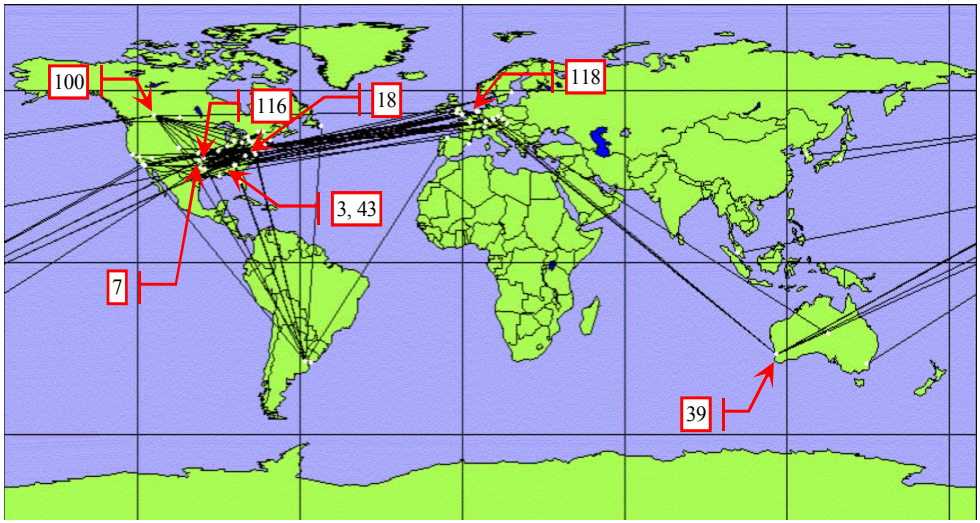
- Caratteristica principale, rispetto al P2P puro : definizione dinamica di un livello gerarchico nella rete
  - UltraPeers: eseguono tutte le funzioni del protocollo di Gnutella (ping, pong, query, queryhit, push). Formano un overlay Gnutella e costituiscono un proxy verso la rete per i leafnodes gestiti
  - Leafnodes: Si connettono agli UltraPeers per utiliazzare la rete Gnutella. Non partecipano alla propagazione di query/ping

- Struttura della rete Hub based network



TOPOLOGIA DI UNA RETE P2P IBRIDA





Rete Gnutella (222 nodes).
Sulla destra è mostrata la collocazione fisica dei rilevata il giorno 01.08.2002

I numeri corrispondono ai numeri dei nodi mostrati nella rete astratta sulla sinistra

- La rete virtuale non corrisponde a quella fisica.
- La struttura dei Ultrapeer (hub strucure) è chiaramente rilevabile dalla rete astratta

GNUTELLA 0.6: OBIETTIVI

- Gnutella 0.6: soluzione di compromesso rispetto alla versione 0.4
  - diminuzione dei messaggi scambiati
  - affidabilità paragonabile (non esiste un unico punto di centralizzazione)

- gli host con una ridotta capacità di calcolo/banda non fanno parte dell'overlay Gnutella, anche se possono accedervi
  - obiettivo: proibire ad host particolarmente 'lenti' di rallentare la ricerca di peer/risorse
  - nello stesso tempo a nessun peer è proibito di utilizzare i servizi di Gnutella
- idea alla base di diverse altre implementazioni di sistemi P2P (Kazaa,JXTA,..)

- Breve Storia: Gnutella 0.6 sviluppato a partire da Gnutella 0.4 nel 2001. Da allora:
  - disponibili diverse implemetazioni (Limewire, Bearshare,…)
  - definite diverse ottimizzazioni (privacy, scalability, performance,…)

GNUTELLA 0.6: TIPI DI PEER

L'architettura prevede due tipi di nodi

- UltraPeer: costituiscono un proxy per i rispettivi nodi foglia
  - 10-100 connessioni con i nodi foglia, < 10 connessioni con altri ultrapeer
  - deve possedere opportune caratteristiche (es: connessione veloce ad Internet nonbloccate da firewall)
- Nodi Foglia
  - non accettano connessioni Gnutella
  - aprono solo alcune connessioni verso nodi SuperPeer (spesso solo una)
  - una foglia può chiedere dinamicamente di diventare SuperPeer
- L'overlay Gnutella 0.6 assume caratteristiche simili ad Internet: nodi caratterizzati da bassa banda sono connessi ai routers che trasmettono i dati su collegamenti a larga banda (backbones)

GNUTELLA 0.6: ELEZIONE DI ULTRAPEERS

- UltraPeer: eliminano il carico della gestione del routing delle queries dai leaf nodes
  - routing delle queries e dei messaggi di ping: avviene solo tra UltraPeers
- Ogni host determina in modo autonomo, prima di connettersi alla rete Gnutella, se possiede le caratteristiche necessarie per diventare un superpeer. In questo caso l'host si qualifica come UltraPeer Capable altrimenti come LeafNode
- Esempio:
  - Limewire, richiesti 10kB/s per UpLoad, 20kB/s per DownLoad per poter essere eletto UltraPeer
- Non esiste un server centrale che confronta le capacità dei diversi peer e decide quali di essi diverranno Ultra Peers
- E' definito un algoritmo distribuito il cui scopo è quello di definire dinamicamente il numero di Super peers utili per l'overlay
- UltraPeer Dynamic Tuning: esempio di self organization

ULTRAPEER CAPABILITY

Criteri per la determinazione dei nodi UltraPeer Capable:

- Assenza di firewall/NAT
  - Approssimazione: verifica se il nodo ha ricevuto connessioni in ingresso
- Larghezza di Banda
  - Approssimazione: calcolo del throughput di download/upload
- Quantità di RAM disponibile per le tabelle di routing
- OS Adatto. Alcuni sistemi operativi gestiscono un alto numero di sockets in modo più efficiente rispetto ad altri. Limewire vieta ad un node di divenire SuperPeer se sul nodo è installata una vecchia versione di sistema operativo
- Potenza di calcolo (ciclo di clock della CPU) per gestire ed inoltrare le query in arrivo
- Uptime futuro
  - Euristica: l'uptime futuro è proporzionale a quello passato

PRINCIPI DI ELEZIONE DEGLI ULTRAPEER

- Un nodo UltraPeer capable può diventare UltraPeer se vi è necessità di ulteriori UltraPeer nell'overlay
- La necessità di UltraPeer può essere stimata considerando il numero di UltraPeer attualmente presenti sull'overlay
- Tali dati possono essere comunicati al momento dell' handshake
- Necessità di nuovi header
  - ricordiamo la struttura generale dell'header:

X-Header-Name: $headervalue_1$, $headervalue_2$

GNUTELLA HANDSHAKE

- Le informazioni per l'elezione degli ultrapeer sono scambiate durante la sequenza di handshaking
- Header introdotti in Gnutella 0.6 a questo scopo
  - **X-UltraPeer=true**, segnala che il peer è ultrapeer capable
  - **X-UltraPeerNeeded**, usato per bilanciare il numero di ultrapeers
  - **X-TryUltraPeer** come X-try di Gnutella 0.4, ma contiene solo indirizzi di UltraPeers
  - **X-Degree** numero di foglie gestite da un UltraPeer
  - **X-QueryRouting**, segnala il supporto per QueryRouting Protocol (QRP)
- Un nodo Gnutella 0.6 può cambiare il proprio stato (Leaf Node/SuperPeer) durante la propria vita

UNA FOGLIA SI CONNETTE AD UN SUPERPEER

Nodo A

GNUTELLA CONNECT/0.6

User-Agent: LimeWire 1.9

X-Ultrapeer: False

X-Query-Routing: 0.1

X-My-Address: 10.254.0.16:6349

UltraPeer B

GNUTELLA/0.6 200 OK

User-Agent: LimeWire 1.9

X-Ultrapeer: True

X-Ultrapeer-Needed: false

X-Try-Ultrapeers: 23.35.1.146:6346,
X-Try: 24.37.144:6346,193.205.63.22:6346

X-Query-Routing: 0.1

X-My-Address: 10.254.0.16:6346

GNUTELLA/0.6 200 OK

GNUTELLA/0.6 200 OK

UNA FOGLIA SI CONNETTE AD UN SUPERPEER

- Al termine dell'esempio precedente
  - la foglia è coperta dal SuperPeer
  - La foglia elimina eventuali connessioni a non SuperPeer
  - La foglia invia una tabella di routing al proprio SuperPeer (QRP, Query Routing Protocol)

UNA FOGLIA SI CONNETTE AD UN'ALTRA FOGLIA

Nodo A

GNUTELLA CONNECT/0.6

X-Ultrapeer: False

Nodo B

GNUTELLA/0.6 503 I am a shielded leaf node

X-Ultrapeer: False

X-Try-Ultrapeers: 18.2.3.14:6346,18.1.17.2:6346

- B rifiuta la connessione da A e "ridirige" A verso altri ultrapeer di cui fornisce indirizzo IP e porta
- A tenterà di stabilire una connessione con essi
- i nodi foglia
  - non accettano connessioni Gnutella da parte di altri nodi foglia
  - il nodo foglia poteva essere in precedenza un UltraPeer e poi ha modificato il suo ruolo, ma la modifica non è ancora stata notificata al server di bootstrap/sulla rete

CONNECTION HEADERS

- **X-Ultrapeer:**
  - True: l'host è un UltraPeer,
  - False: l'host è is una Foglia
- **X-Ultrapeer-Needed:**
  - True: l'host vuole aprire ulteriori connessioni con altri UltraPeer
  - False: l'host è disposto ad aprire ulteriori connessioni con foglie, ma non con altri UltraPeeers
- **X-Degree:** rappresenta il numero di connessioni che un ultrapeer è disposto ad aprire con altri UltraPeers.
- **X-Try-UltraPeers:** una lista di coppie (ndirizzo IP -porta) scambiati in fase di connessione

CONNESSIONE ULTRAPEER/ULTRAPEER

Ultrapeer A	Ultrapeer B
GNUTELLA CONNECT/0.6 X-Ultrapeer: True	
	GNUTELLA/0.6 503 Service unavailable X-Ultrapeer: True X-Ultrapeer-Needed: True

- B ha già aperto un grande numero di connessioni
- B rifiuta la richiesta di connessione da A

LEAF GUIDANCE (ABDICAZIONE ULTRAPEER)

Nodo A

SuperPeer B

GNUTELLA CONNECT/0.6

X-Ultrapeer: true

GNUTELLA/0.6 200 OK

X-Ultrapeer: true

X-UltraPeer-Needed:False

GNUTELLA/0.6 200 OK

X-UltraPeer: False

- L'host UltraPeer Capable A si connette all'UltraPeer B che gestisce poche foglie
- B chiede ad A di diventare una sua foglia
  - A dichiara la propria disponibilità a divenire una foglia inviando X-Ultrapeer: False.

LEAF GUIDANCE

Meccanismo di leaf guidance:

- forzare la scelta del ruolo di un nuovo peer che si connette alla rete
- obiettivo: mantenere un bilanciamento tra i nodi foglia e gli UltraPeer
- meccanismo locale, ma se eseguito da tutti gli Ultrapeer l'intera rete tende a bilanciarsi
- implementato mediante la introduzione di nuovi header
  - X-Ultrapeer-Needed:True accetto il ruolo che ti sei scelto come Ultrapeer
  - X-Ultrapeer-Needed:False possiedo già un numero sufficiente di UltraPeer, ma non di nodi foglia. Puoi diventare un nodo foglia?
- Il meccanismo è effettivo quando un peer dichiara che dichiara la sua intenzione di diventare Ultrapeer

LEAF GUIDANCE

Leaf Guidance Refusal

- Un peer che si collega con X-UltraPeer= true, potrebbe di fatto essere già un Ultra Peer, perchè ha già accettato connessioni come UltraPeer
  - connessione precedente accettata da un nodo foglia
  - connessione precedente con un UltraPeer che ha risposto X-UltraPeer needed=true
- In questo caso il peer non può regredire a nodo foglia, perchè non si può comportare come foglia su certe connessioni, come UltraPeer su altre
- Risponde GNUTELLA/0.6 200 OK con X-Ultrapeer:true e rimane un UltraPeer

LEAF GUIDANCE

- Ogni UltraPeer ha un insieme di k slot disponibili per altri ultrapeer e n slot per i nodi foglia
- Se si connette un nodo foglia , la connessione viene accettata se c'e' spazio, altrimenti viene rifiutata
- se si connette un SuperPeer
  - se ci sono abbastanza nodi foglia (percentuale di k) , attiva il leaf guidance
  - altrimenti accettalo come ultrapeer

SUPERPEER PROTOCOL: RIASSUNTO

- Foglia F → SuperPeer S
  - F diventa una foglia di S
  - F rifiuta connessioni da altri peer
  - F invia ad S una QRP routing table

- Foglia F → Foglia G gestita da S: F diventa foglia di S

- SuperPeer S1 → Superpeer S2
  - S2 può rifiutare la connessione da S1 se gestisce già un alto numero di connessioni
  - se entrambi i superpeer gestiscono un numero sufficiente di foglie, entrambi rimangono superpeers e stabiliscono una connessione tra di loro
  - altrimenti uno dei due può diventare un nodo foglia gestito dall'altro

APERTURA DELLE CONNESSIONI

- Un SuperPeer S può decidere di rifiutare la connessione richiesta da un leaf node

- In questo caso S fornisce, al momento dell'handshaking, indirizzi di altri SuperPeers, inviando connection pongs che sono stati ricevuti, in precedenza, dalla overlay network dei super-peers

  Esempio: X-try-SuperPeers:68.37.233.44:9376, ...

- Alcuni connections pongs vengono restituiti anche nel caso in cui la connessione venga accettata. In questo caso le informazioni ricevute possono essere utilizzate in seguito per stabilire connessioni con SuperPeers alternativi

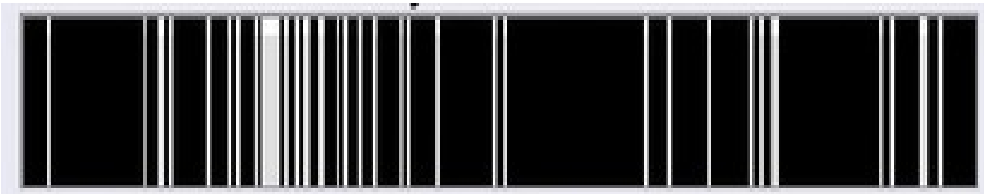
QUERY ROUTING PROTOCOL

Cosa è?

- Governa il routing delle query verso i nodi foglia che hanno maggior probabilità di soddisfare le query
- Questo viene implementato cercando le keywords della query all'interno di una Query Routing Table (QRT)
- Ogni foglia invia al proprio SuperPeer una sua QRT
- QRT del SuperPeer costruita a partire dall QRT dei suoi nodi foglia

QUERY ROUTING PROTOCOL

Query Routing Table = Vettore R di 64K,contiene informazioni sui file condivisi dalle foglie



- Ogni risorsa è individuata da un insieme di parole chiave
- Hash del nome del file e delle parole chiave che lo identificano.
- Il risultato della applicazione della funzione hash è un valore v compreso tra 0 e 65536. R[v]=1.
- Routing table = Bitvector. non memorizza il valore delle parole chiave, solo un insieme di flags che indicano la presenza dei valori delle chiavi

QUERY ROUTING PROTOCOL

L'host A condivide un singolo file individuato da diverse parole chiave, ad esempio application, software,JAVA,...

La routing table ha lunghezza 8

La funzione hash viene applicata al nome del file ed a tutte le parole chiave

h('application') = 2, h('software')=5, h('JAVA')=7

routing table di A= bitmap 00100101

QUERY ROUTING PROTOCOL

- Risparmio di spazio a spese della precisione: è possibile che la funzione hash applicata a stringhe diverse restituisca la stessa posizione nel bitvector.

- Data una query Q
  - se tutte le posizioni del bitvector corrispondenti agli hash dei valori di contenuti in Q sono 0, la risorsa ricercata non è sicuramente condivisa dal nodo
  - altrimenti è possibile che il nodo possegga quella risorsa, ma non è certo
- Possibilità di falsi positivi, mentre non esistono falsi negativi

Bloom Filters = Struttura dati utilizzata per rappresentare in modo efficiente un insieme di oggetti. Si basa su tecniche probabilistiche.

- Bitvector di Gnutella 0.6= Versione semplificata di Bloom Filter

ROUTING DELLE QUERY: VERSIONE BASE

- La QRT viene compressa, suddiviso in blocchi ed inviato da un leaf node al proprio UltraPeer che lo memorizza e lo utilizza come Query Routing Table(QRT)

- Query Routing

  - Un LeafNode invia una richiesta al proprio UltraPeer

  - L'UltraPeer
    - invia la query ad un Leafnode L solo se è possibile che L soddisfi la query. Controllo effettuato mediante la QRT di L.

    - applica le stesse funzioni hash applicate dal leaf node ad ogni parola chiave della query

    - TTL enhanced flooding: invia la query ad altri UltraPeer peers per individuare altri hosts che condividono il file richiesto

ROUTING DELLE QUERIES: VERSIONE BASE

- Routing dei query hits:
  - Quando un LeafNode riceve una richiesta, controlla di possedere effettivamente il file richiesto
  - In caso positivo, il Leafnode invia un query hit al proprio UltraPeer
  - Il messaggio viene instradato all'indietro lungo lo stesso cammino che aveva percorso la query (backward routing)
- Scambio dei files:
  - Avviene direttamente tra i nodi interessati, tramite connessioni HTTP.

GNUTELLA 0.6: ROUTING INDEXES

- Combinazione delle tabelle: i SuperPeer possono combinare le QRT proprie e dei propri Leaf Node calcolando l' or bit a bit delle posizioni corrispondenti nelle diverse QRT.
- ogni superpeer spedisce la routing table combinata ai propri vicini
- Last Hop Saving: non inviare una query ad un vicino se TTL = 1 e il vicino non ha nessun Leaf Node che possiede la risorsa ricercata (bit=0 in tutte le posizioni corrispondenti a parole chiave che individuano la risorsa)
- Estensione di questa soluzione (utilizzata in alcuni client Gnutella)
  - Propagare le proprie routing table nella rete entro un certo raggio,determinato dal TTL
  - Modificare il contenuto della QRT in modo che ogni sua posizione indichi la distanza dall'eventuale host che possiede il file corrispondente

ROUTING INDEXES: SOLUZIONE ALTERNATIVA

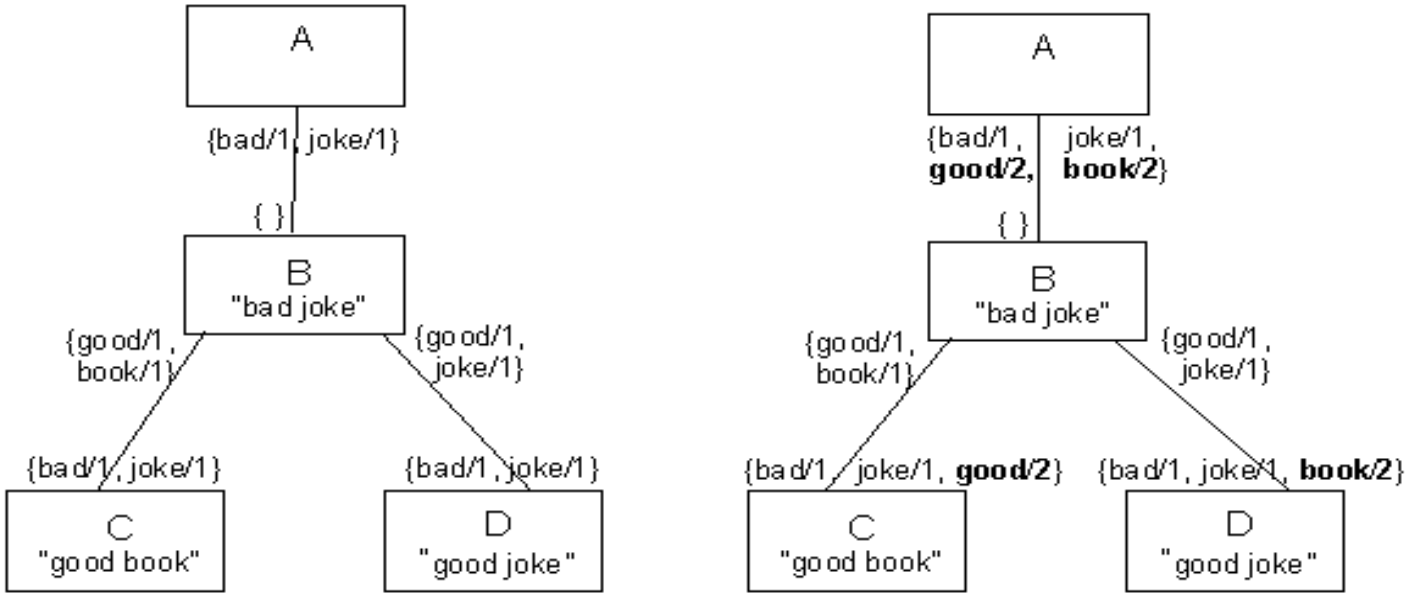


Tabella di routing di D [∞, 2, ∞, 1, ∞, 0,∞,0...], con

hash('book')=1, hash('bad')=3,hash('good')=5,hash('joke')=7

il valore ∞ (infinity)indica che nessun peer nel raggio definito dal TTL possiede un file identificato dalla chiave k corrispondente alla posizione di ∞ nel vettore

Scambio continuo delle tabelle di routing con i peer vicini

ROUTING INDEXES: SOUZIONE AVANZATA

- Routing tables = vettore di interi
- Una tabella ROUC distinta per ogni connessione C con i peer vicini
- ROUC[h]= distanza minima, in numero di hops lungo quella connessione, da un host che possiede una risorsa descritta da una chiave k tale che hash(k)=h.
- Gli hosts scambiano periodicamente le proprie routing tables con i vicini ed aggiornano le proprie routing tables con l'informazione ricevuta
- Supponiamo che l'host X_i sia connesso con gli hosts $\{Y_1,...,Y_m\}$ e sia $ROUY_j$ la tabella ricevuta da Y_j. La tabella di routing ROUX di X_i è aggiornata nel modo seguente: per ogni posizione h della tabella di routing
  - se X_i posside un file identificato da almeno una parola chiave k tale che hash(k) = h, allora ROUX[h]=0
  - altrimenti $ROUX[h]= \min_{j \neq i} (ROUY_j[h])$
- Se TTL >0, la tabella di routing risultante viene propagata ai propri vicini incrementando di 1 il valore di ogni posizione

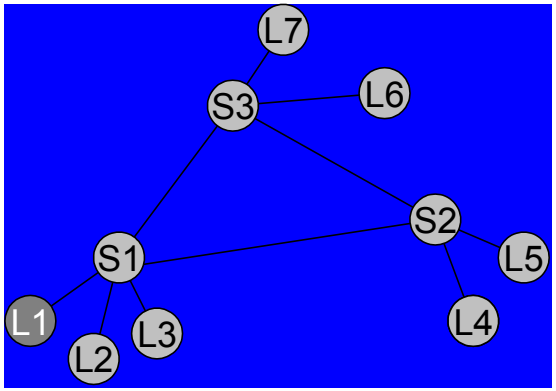
GNUTELLA 0.6: ALTRI MESSAGGI

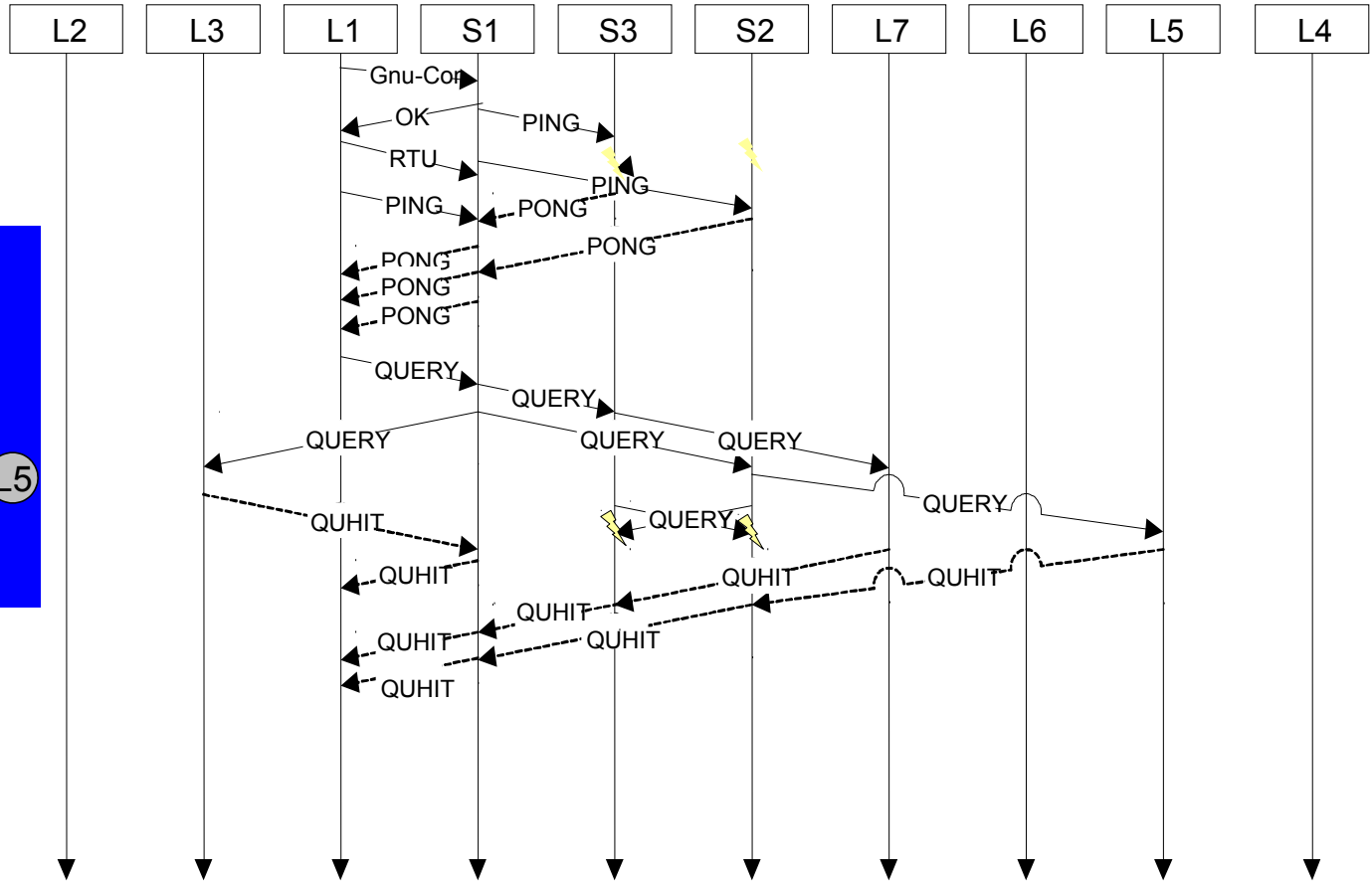
- Richiesta di files (Content Request)
  - QUERY (gli stessi di Gnutella 0.4)
  - QUERY_HIT (gli stessi di Gnutella 0.4)
- Keep alive:
  - PING (gli stessi di Gnutella 0.4)
  - PONG (gli stessi Gnutella 0.4)

GNUTELLA 0.6: ESPLORAZIONE DELLA RETE

- I messaggi di ping pong vengono scambiati solamente tra i SuperPeer
- Un SuperPeer non propaga mai un messaggio di ping ai propri LeafNodes
- In questo modo ogni SuperPeer "mette al riparo" i propri LeafNodes dal traffico di keep-alive
- Un SuperPeer può ricevere un ping da un proprio LeafNode. In questo caso invia un pong con le informazioni ricevute da altri SuperPeers.
- Il LeafNode può utilizzare questa informazione nel caso in cui il proprio SuperPeer si disconnetta, oppure nel caso in cui voglia aprire connessioni con più SuperPeers.

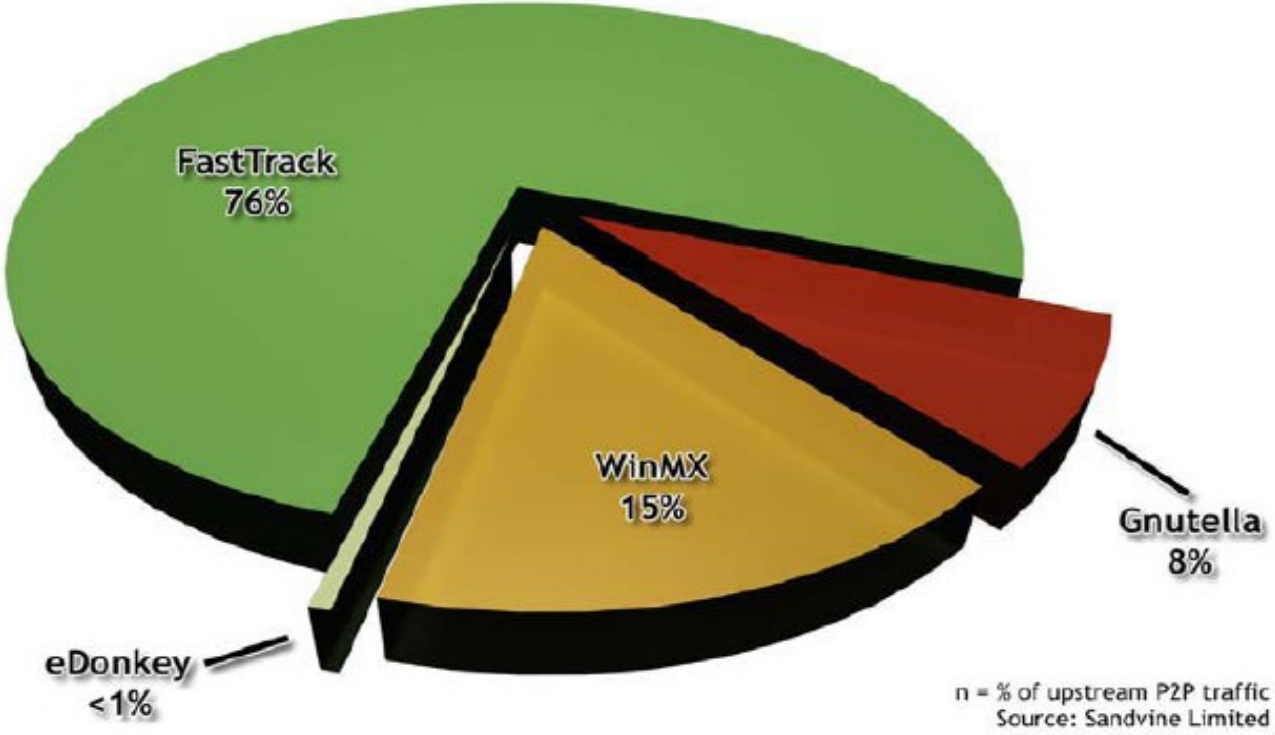
GNUTELLA 0.6: MESSAGGI DI CONTROLLO





KAZAA: CARATTERISTICHE GENERALI

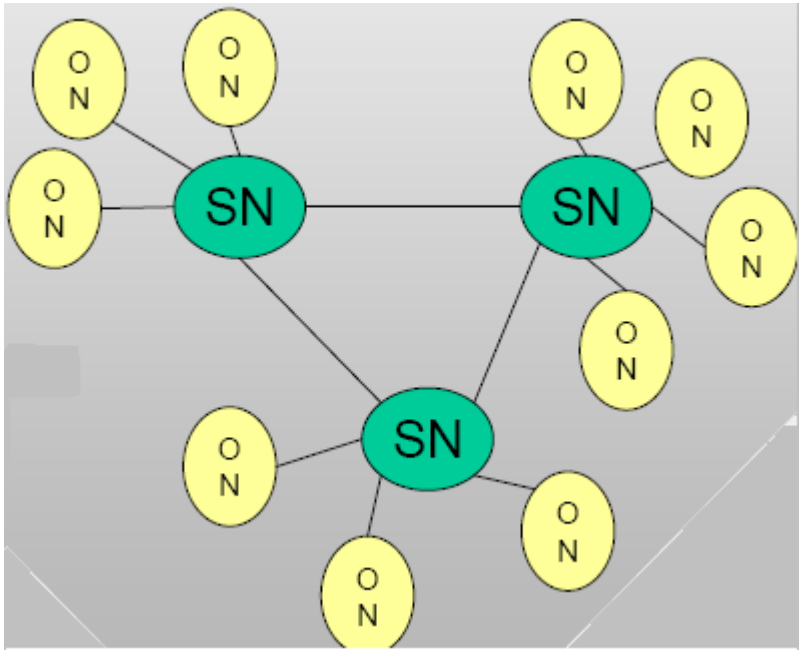
- Sistema P2P proprietario (2001), tutto il traffico è crittato (scarsa documentazione)
- Utilizza il protocollo FastTrack
- Grande diffusione (dati: USA, 2003)



KAZAA: STORIA

- Kazaa creata nel marzo 2001, Niklas Zennstrom
- Consumer Empowerment (Compagnia Olandese)
  - Joltid, Fasttrack, Joost, Skype
- Novembre 2001, costretti ad abbandonare l'Olanda a causa di una condanna
- Trasferimento in Vanatu, compagnia Sharman Networks
- Marzo 2002, vinto il ricorso
- Altre condanne in Australia, USA
- Giudizio finale giugno 2006: pagamento di $100M, Kazaa viene trasformato in un servizio a pagamento
- Dall'agosto 2012 non più attivo

KAZAA: CARATTERISTICHE GENERALI



- Two Tier Architecture (simile a Gnutella 0.6)
  - Definisce due tipi di nodi, Ordinary Nodes(ON), Super Nodes (SN)
  - SN determinati in base a banda, capacità di calcolo, mancanza NAT firewalls
  - In ogni istante, un ON è connesso ad un solo SN (connessione TCP)
  - I SN formano un'overlay network (connessioni TCP)

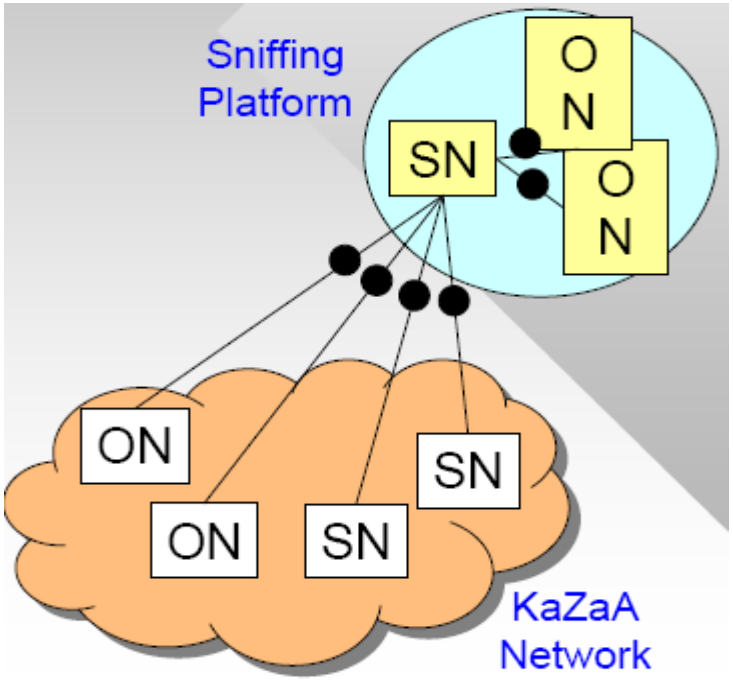
KAZAA: I SUPER NODES

- ON comunica al proprio SN i metadati che descrivono le informazioni che condivide

- Metadata
  - nome, dimensione del file
  - content Hash (utilizzato per identificare il file univocamente)
    - i primi 16 bytes contengono l'MD5 dei primi 300k del file
    - ultimi 4 bytes sono l'hash della lunghezza del file
    - utilizzato per identificare univocamente il file e per riprendere download precedentemente interrotti di file
  - descrittore del file (artista, album, anno,...)

- Responsabilità dei Super Nodes
  - connettere gli ON all'overlay Kazaa
  - mantenere l'overlay
  - tenere traccia solo dei metadati relativi ai propri ON
  - rispondere alle query ed inoltrarle sull'overlay

KAZAA SNIFFING PLATFORM



Caratteristiche dell'esperimento

- si definisce una 'sniffing platform' che include 3 hosts del Politecnico di New York
- all'inizio tre hosts vengono configurati come ON
- attesa che uno degli ON venga 'promosso' a SN da Kazaa
- i due ON vengono forzati a diventare figli del SN appartenente alla piattaforma (modifica registry di windows)
- si analizza il traffico in entrata ed in uscita del SN

ANALISI SPERIMENTALE DELL'OVERLAY

Scopo dell'analisi dell'overlay Kazaa

- quanti SuperNode esistono in media un overlay Kazaa?
- quanti ON sono gestiti in media da un Super Node?
- cosa contiene l'indice di un SuperNode?
- quale è la durata media di una connessione SN/SN o ON/SN?
- come avviene la scelta di un SN da parte di un ON?
- Come vengono gestiti NAT/firewalls?

KAZAA: SUPERNODE LISTS

- Ogni nodo mantiene una cache di SN (SN cache list)
  - ON possiede una lista di alcune centinaia di SN
  - SN possiede una lista di alcune migliaia di SN

- esistono default Server che possono essere interrogati se la cache è vuota

- i nodi Kazaa si scambiano frequentemente le liste di supernode
  - scambio di liste di supernode a livello di handshake SN/SN ed ON/SN (simile ad XtryUltraPeers in Gnutella 0.6)
  - shuffling delle connessioni consente di creare una rete dinamica che può essere descritta come un grafo random
  - comportamento gossip

KAZAA: INGRESSO NELLA RETE

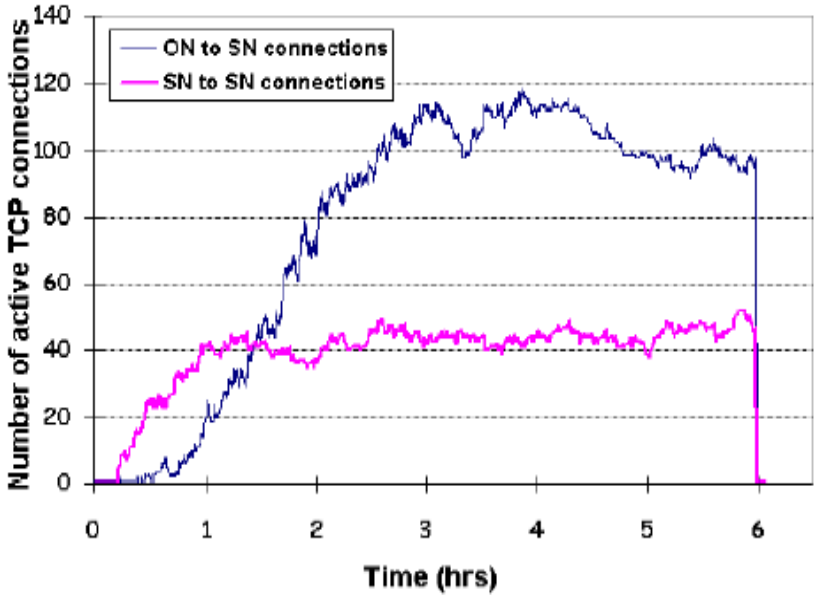
- SN Probing:
  - all'inizio un ON prova ad inviare un pacchetto UDP a tutti i SN nella sua cache list
  - al passo successivo, apre in parallelo più connessioni TCP con alcuni SN selezionati
  - alla fine sceglie un SN e si disconnette dagli altri
  - refresh della supenode list: al momento della connessione ogni SN invia al proprio ON una lista contenente altri 200 SN.

- SN Jumping; ON cambiano frequentemente SN durante la propria permanenza sulla rete
  - possono cambiare SN anche durante una singola ricerca
  - mantengono l'ultima connessione stabilita fino alla ricerca successiva
  - ogni volta manda i propri metadata al nuovo SN contattato
  - un SN cancella i metadata degli ON che si disconnettono

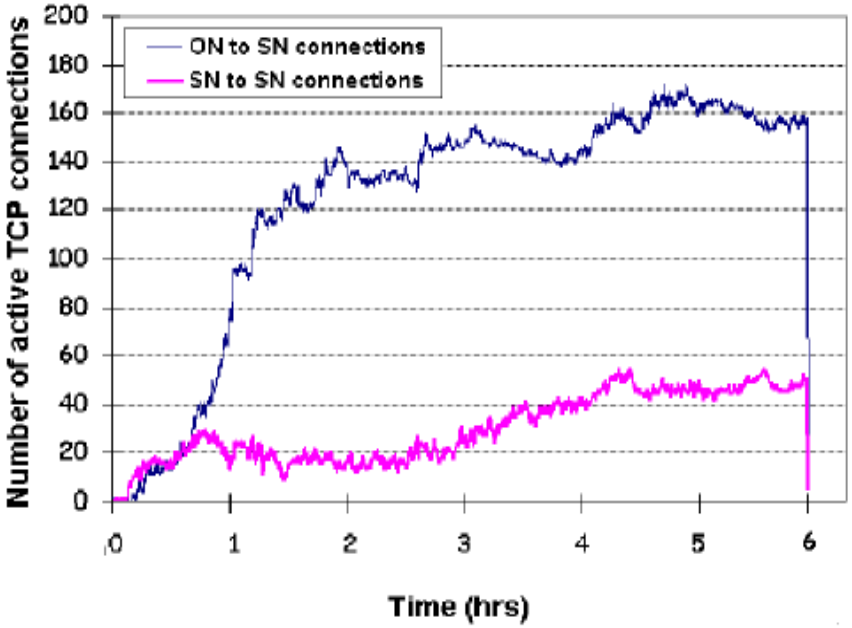
KAZAA: LOOK UP DEI DATI

- ON invia la query al proprio SN (Esempio: ricerca di qualsiasi file che contenga la keyworld "Sting"). La query contiene una lista di parole chiave.
- Le risposte ricevute da SN contengono metadati che descrivono la risorsa individuata ed identificano il peer che offre la risorsa
- Il download è diretto e utilizza il content hash per individuare il file (GET ContentHash)
- Per aumentare le prestazioni il download scarica il file a frammenti dall'insieme di peer che lo condividono
- Se il download fallisce si ricerca nuovamente il file utilizzando il content hash

STRUTTURA DELL'OVERLAY



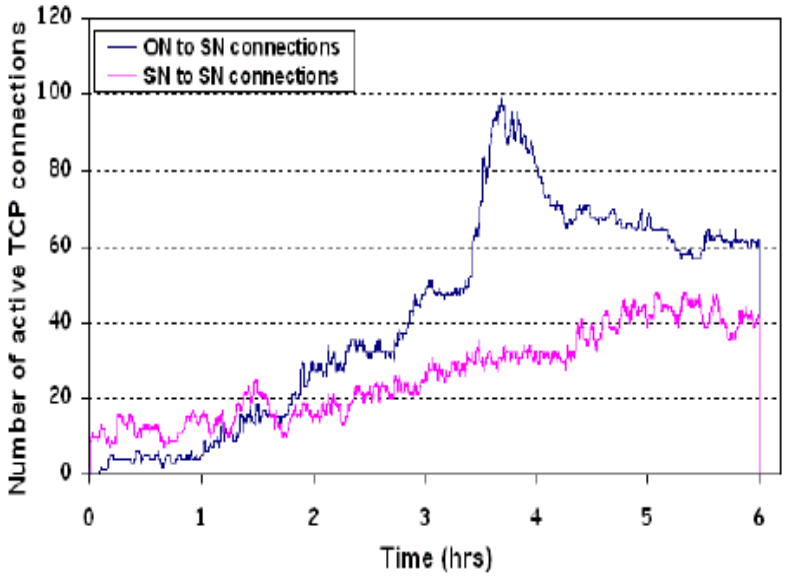
(a) Polytechnic campus - session evolution, Aug. 22, 2003



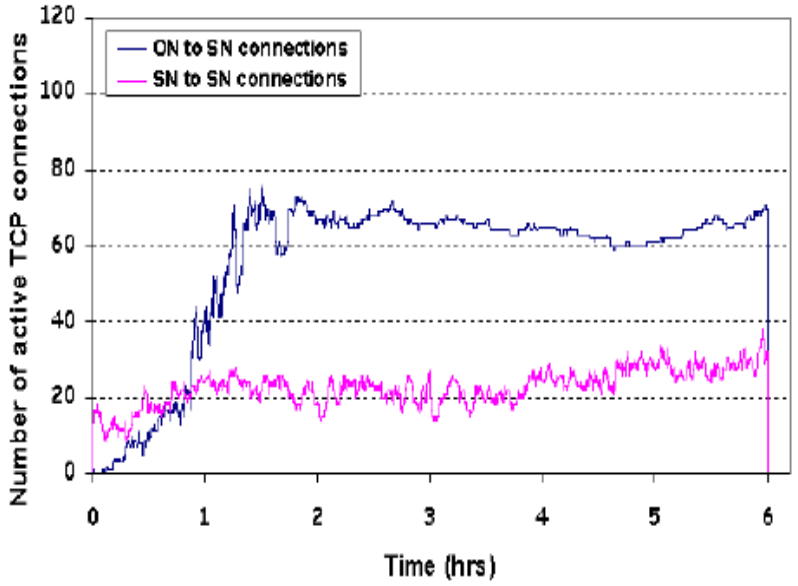
(b) Polytechnic campus - session evolution, Oct 24, 2003

- Valutazione del numero di connessioni simultanee tra il SN sniffer (installato al politecnico) ed altri ON e SN
- Esperimento ripeturo in diversi giorni

STRUTTURA DELL'OVERLAY



(c) Residential Cable Modem - session evolution, Aug. 23, 2003



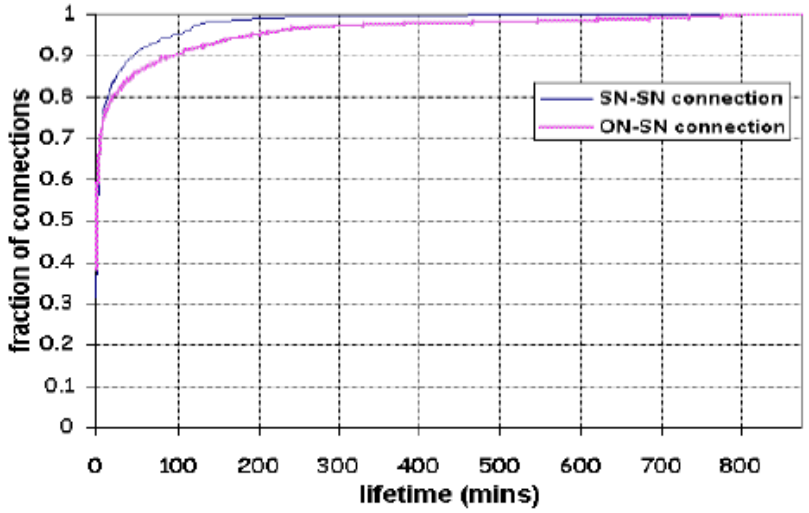
(d) Residential Cable Modem - session evolution, Oct 25, 2003

- Valutazione del numero di connessioni simultanee tra il SN sniffer (installato al politecnico) ed altri ON e SN
- Esperimento ripeturo in diversi giorni

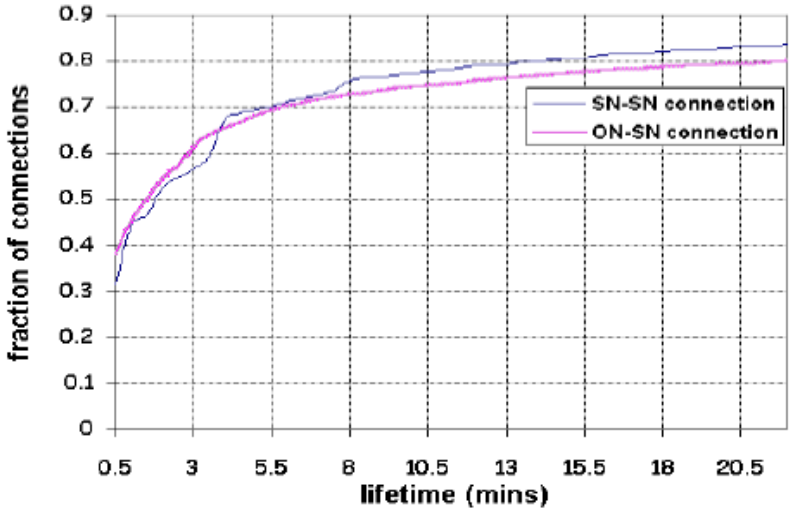
STRUTTURA DELL'OVERLAY

- Il numero di connessioni tra il SN sniffer e gli altri peer (ON e SN) cresce fino ad una certa soglia e poi oscilla intorno a questa soglia
- Il numero di connessioni con gli ON è molto più alto delle connessioni con i SN
- In media 40-50 connessioni SN-SN, circa 100 connessioni SP- ON
- Numero medio di utenti Kazaa: 3 milioni
  - ogni ON mantiene una sola connessione verso il proprio SN
  - nella rete ci sono approssimativamente 3000000/100= 30000 SN
  - ogni SN si connette approssimativamente allo 0.1% degli altri SN nella rete
- Caratteristica 1: topologia della rete dei SN molto sparsa, non è un mesh

DINAMICA DELL'OVERLAY



(a) full duration plot



(b) close-up of the plot

- ascisse: durata della connessione in minuti (800 minuti = 13 ore)

- ordinate: CDF (cumulative distribution function), percentuale di connessioni che hanno durata minore o uguale al valore corrispondente sull'asse delle ascisse

DINAMICA DELL'OVERLAY

Risultati:

- il numero di connessioni simultanee stabilite dal SN sniffer con gli altri SN si attesta intorno ad una soglia, dopo un certo intervallo di tempo, ma....

  le singole connessioni stabilite dal SN variano molto frequentemente

- Durata media delle connessioni SN-SN 11 minuti, delle connessioni SN-ON

  34 minuti, ma esiste un numero molto alto di connessioni che vengono aperte per pochi secondi

- si può anche osservare che se una connessione dura per più di 30 minuti, poi la connessione rimane stabile (56.6 minuti in media per le connessioni ON-SN e 23.3 minuti per le connessioni SN-SN)

DINAMICA DELL'OVERLAY SN-SN

- Caratteristica 2: L'overlay tra i SN è altamente dinamica. Ogni SN cambia le sue connessioni con altri SN molto frequentemente
- Motivazioni:
  - connessioni instaurate per il solo scambio delle liste di Super nodes.
  - ricerca di SN con un basso carico
  - alto livello di shuffling della rete di SN consente di cambiare molto spesso la struttura dell'overlay SN/SN
    - la dinamicità dell'overlay SN/SN consente agli ON gestiti di esplorare una parte più ampia della rete
    - se un ON rimane sulla rete KAZAA per un lungo periodo di tempo, ha la possibilità di spedire le sue query a diversi SN e quindi incrementa le proprie possibilità di individuare le informazioni ricercate

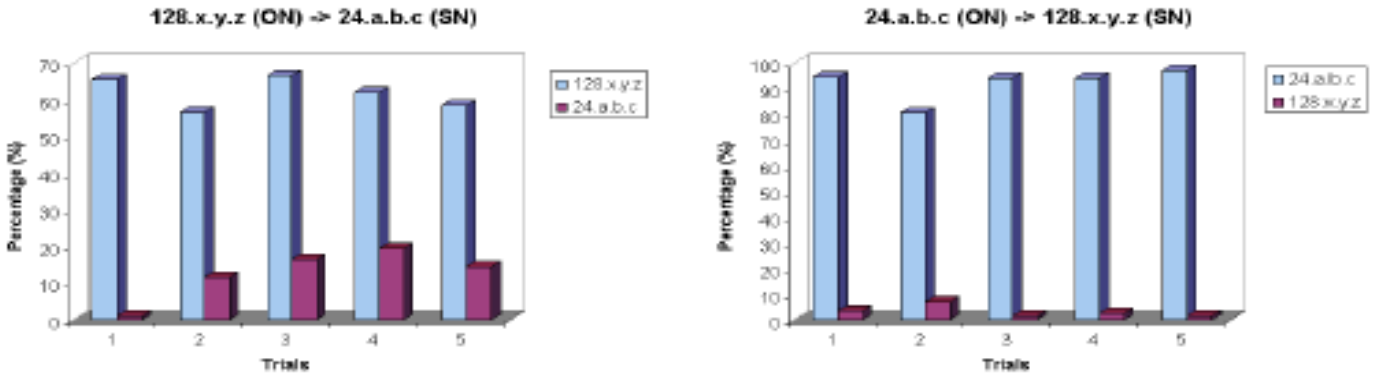
DINAMICITA' CONNESSIONI ON-SN

- Anche le connessioni ON-SN variano frequentemente
- Motivazioni
  - allo start-up un ON invia dei pacchetti UDP ai SN nella lista di SuperNodes
    - poi apre un insieme di connessioni con i SN che hanno risposto ed alla fine sceglie un solo SN a cui connettersi.
    - quindi esistono, al momento dello start-up, un alto numero di connessioni che vengono aperte e poi immediatamente chiuse
  - Ogni ON si disconnette frequentemente dal proprio SN e si connette ad altri SN. Ogni query viene inviata al nuovo SN per la ricerca di nuovi dati

SELEZIONE DEI VICINI: PREFIX MATCHING

- Quando un ON si connette ad un SN, esso restituisce all'ON una lista L di SN
- Come avviene la scelta dei SN da inserire in L?
- Prefix Matching: similarità rispetto agli indirizzi IP. E' il criterio di scelta dei SN da inviare all'ON
  - si valuta la correlazione tra gli indirizzi IP degli SN in L e l'indirizzo IP di ON
  - un alto numero di SN in L hanno prefissi IP 'simili' a quello di ON

SELEZIONE DEI VICINI: PREFIX MATCHING



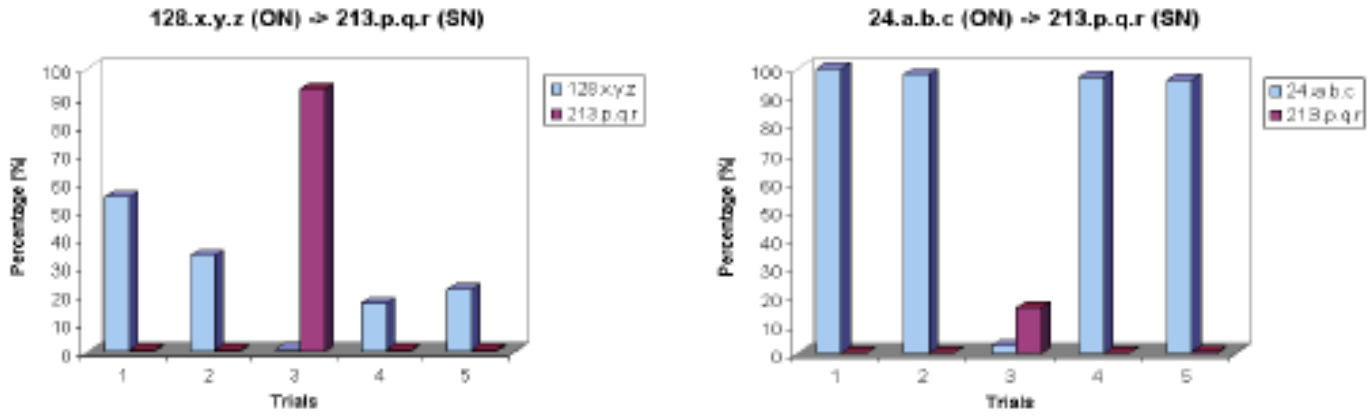
(a) The child ON IP address is from 128/8 and of parent SN is from 24/8

(b) The child ON IP address is from 24/8 and of parent SN is from 128/8

- Correlazione tra l'indirizzo IP dell'ON e gli indirizzi IP dei SN in L

- gli indirizzi ricevuti tendono ad avere prefissi comuni con quello dell'ON
- SN ed ON locati in USA

SELEZIONE DEI VICINI: PREFIX MATCHING



(c) The child ON IP address is from 128/8 and parent SNs are from 213/8

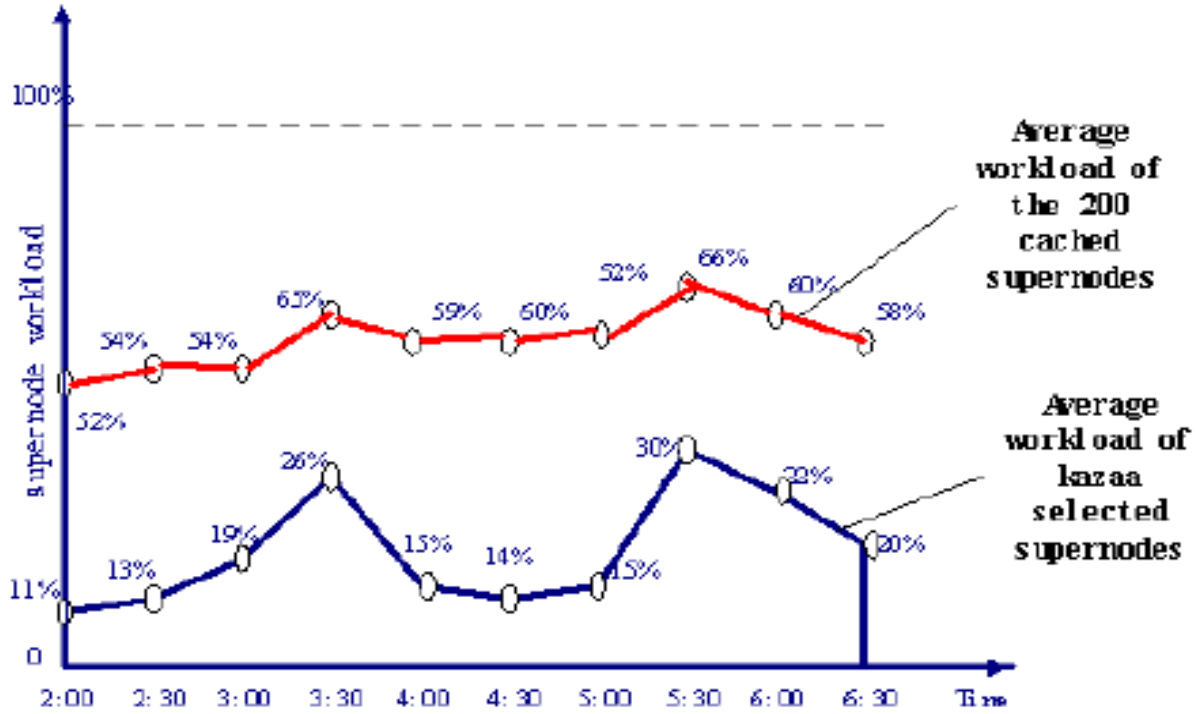
(d) The child ON IP address is from 24/8 and parent SNs are from 213/8

- SN locato in Svezia (indirizzo 231.a.b.c), ON in USA
- In questo caso la località diminuisce
- Il SN possiede meno conoscenza dei SN presenti in USA e non riesce a trovare SN con prefissi che combaciano con quello dell'ON
- La situazione migliora quando l'indirizzo dell'ON e' 24.a.b.c (24.a.b.c rete densa di supernodes)

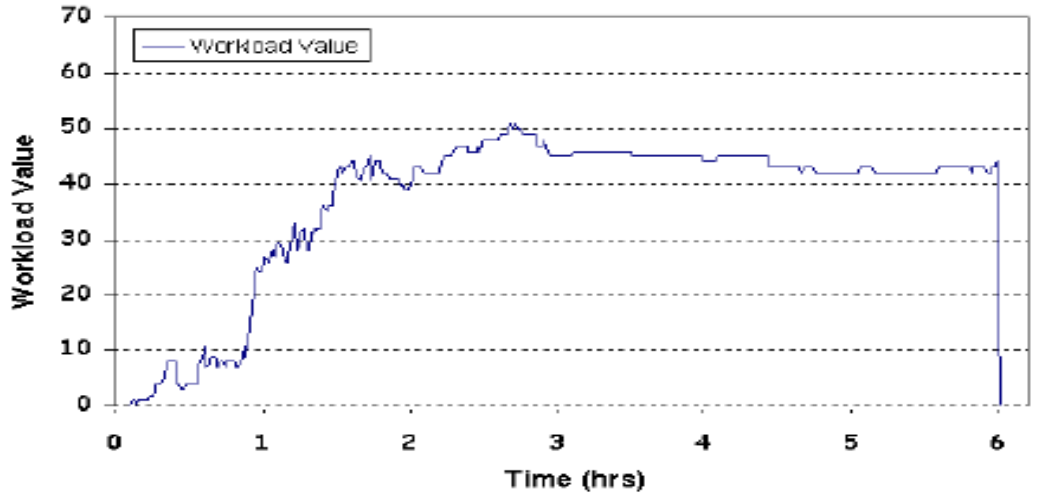
SELEZIONE DEI SN

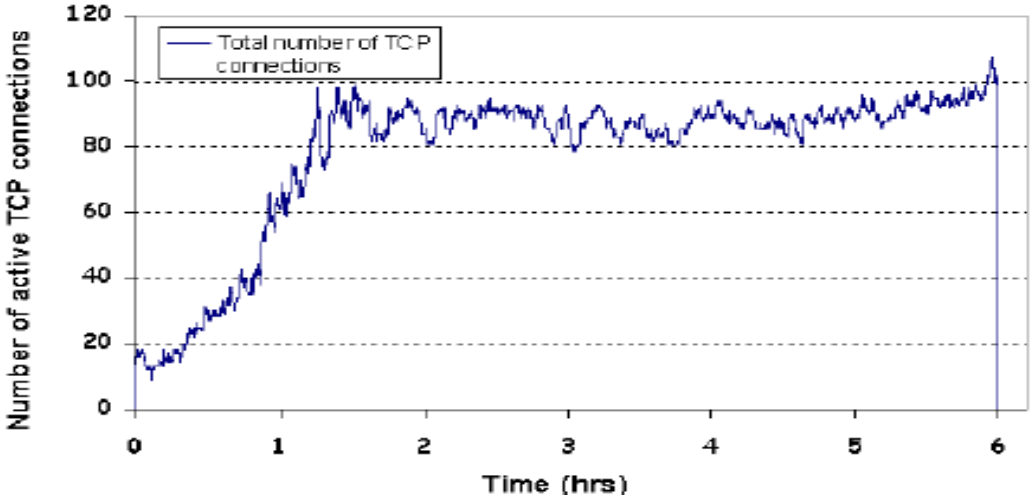
- Proprietà 3: Le connessioni ON-SN ed SN-SN vengono stabilite in base ai seguenti criteri:
  - workload dei SN
  - località
- La valutazione viene effettuata nel seguente modo:
  - ogni ON mantiene una cache in cui registra una lista di SN
  - si forza un ON a connettersi ad un insieme di SN scelti dalla sua cache di SN
  - L'ON poi sceglie un sottoinsieme dei SN a cui si è connesso.
  - si calcola il rapporto del workload dei SN scelti rispetto a quello di quelli scartati
  - per ogni SN vengono registrati: indirizzo IP, porta e workload
- Workload: gli autori non sono riusciti a determinare precisamente cosa sia il workload di un nodo, ma si può ipotizzare che esso sia collegato al numero di connessioni gestite, o al traffico gestito dal SN
- Risultato: Un ON/SN sceglie preferibilmente un SN caratterizzato da un basso workload

SELEZIONE DI SN



CARATTERIZZAZIONE DEL WORKLOAD





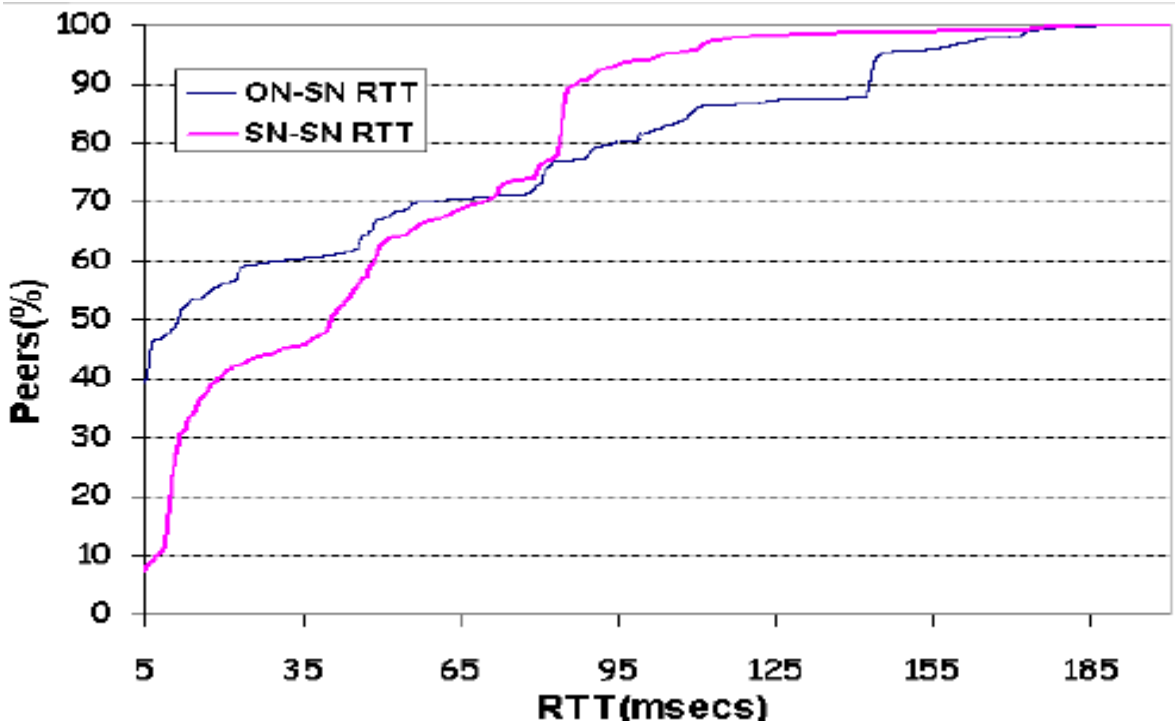
Esiste una forte correlazione tra

- workload di un SN
- numero di connessioni gestite da quel SN

SELEZIONE DEI VICINI: LOCALITA'

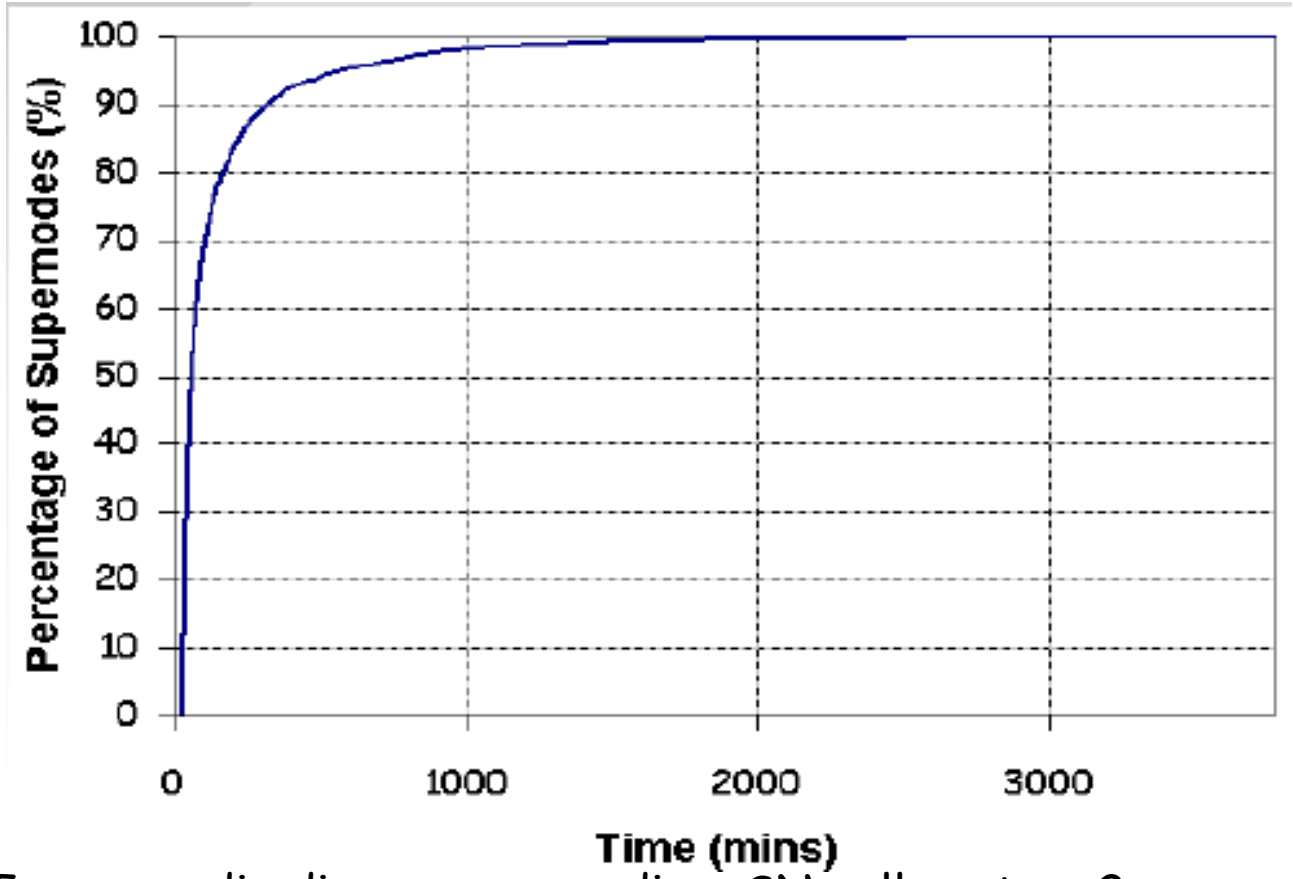
- La località è un altro criterio che viene utilizzato per la scelta dei vicini sia dai SN che dagli ON
- Località misurata in base:
  - RTT(Round Trip Time): misura del tempo impiegato da un pacchetto di dimensione trascurabile per viaggiare da un host della rete ad un altro e tornare indietro.
  - Prefix Matching: Similarità rispetto ai prefissi degli indirizzi IP
    - viene valutata la correlazione tra gli indirizzi IP di due host tra cui viene stabilita una connessione
- Kazaa tiene in considerazione questi aspetti quando crea dinamicamente l'overlay
  - il traffico viene confinato all'interno di uno stesso dominio
  - la ricerca risulta essere localizzata all'interno di un dominio

SELEZIONE DEI VICINI: RTT



- Sull'asse delle Y percentuale di peer vicini (SN ed ON) la cui connessione presenta un RTT minore o uguale al valore riportato sull'asse delle x
- Il 60% delle connessioni SN-SN presenta un RTT < 50 ms
- Il 40% delle connessioni ON-SN presenta un RTT <5 ms

PERMANENZA MEDIA DI UN SN SULL'OVERLAY



Tempo medio di permanenza di un SN nella rete = 2 ore e mezzo

NAT E FIREWALL TRASVERSAL

- Meccanismi utilizzati:
  - Porte dinamiche
  - Connection reversal
- I primi client FastTrack utilizzavano una porta fissa per le connessioni TCP (porta 1214)
- In questo modo gli amministratori di rete potevano facilmente configurare i firewall per individuare il traffico P2P e proibire le connessioni tra peers Kazaa
- Per questa ragione,i più recenti client Kazaa, determinano dinamicamente la porta tramite cui accettano le connessioni. Quando un ON stabilisce la connessione con il suo SN, informa il SN sulla porta che sta utilizzando
- Questo meccanismo rende più complessa l'individuazione del traffico Kazaa

CONCLUSIONI

- Kazaa può essere considerata una delle applicazioni P2P di maggior successo riprende molte idea da Gnutella 0.6
- Caratteristiche principali di Kazaa possono essere considerate delle linee guida per la progettazione di overlay P2P non strutturati
- Caratteristiche principali:
  - Rete completamente distribuita
  - Sfrutta l'eterogeneità degli hosts
  - Bilanciamento del Carico: scelta dei SN in base al loro workload
  - Sfrutta la località
    - vicinanza dei nodi nell' overlay in termini di latenza e di topologia della rete
    - Vantaggio: riduce il tempo di risposta ad una query, confina il traffico all'interno di AS
    - Svantaggio: limita la ricerca, la ricerca tende a rimanere limitata all'interno di un AS

CONCLUSIONI

- Connections Shuffling: le connessione che definiscono l'overlay vengono continuamente 'rimescolate'
  - filosofia 'gossip'
  - consente la ricerca in parti diverse della rete
  - permette di trovare peers alternativi se un peer si disconnette durante il download
  - facilita il download da altri SN se il proprio SN si disconnette durante il download del file

- Algoritmi di Gossiping efficienti: utilizzati per scambiare le liste di SN sull'overlay. Consentono di implementare il 'connection shuffling'

- Firewalls e NAT:
  - Inversione del senso della connessione
  - Modifica della porta di ascolto