Lezione n.3
Sistemi P2P Ibridi
Gnutella 0.6- Kazaa
Laura Ricci
3/3/2009

documentazione per questa lezione:

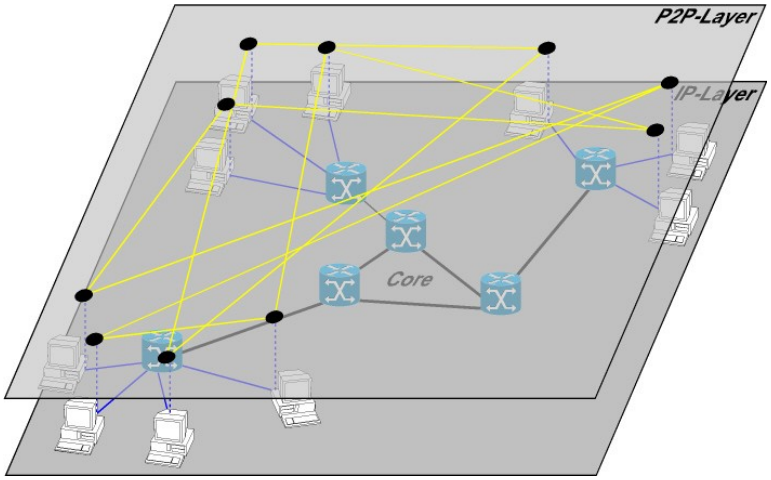
sulla pagina del corso:articolo su Kazaa

+

http://wiki.limewire.org/index.phptitle=How_Gnutella_Works

RIASSUNTO DELLA PRESENTAZIONE

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

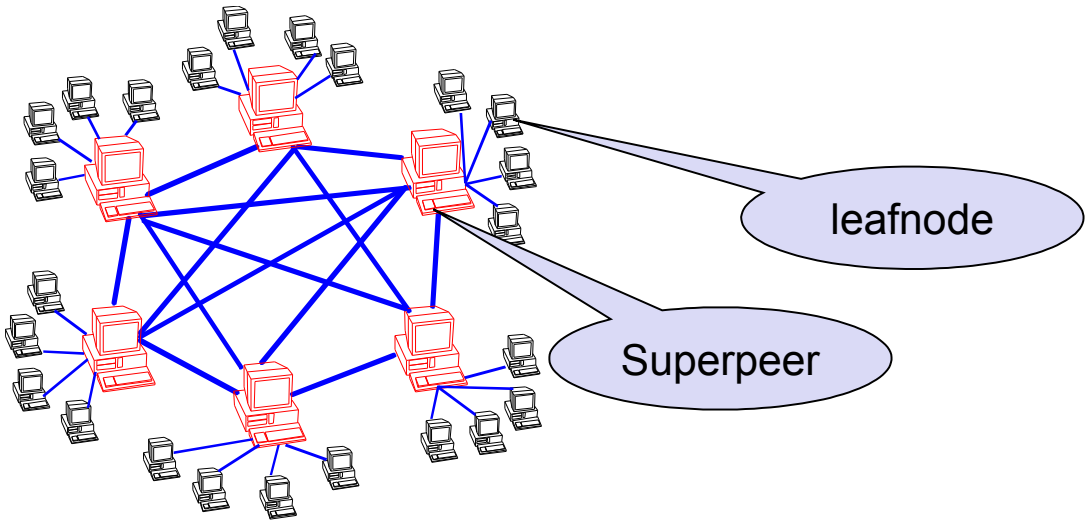


GNUTELLA 0.6: CARATTERISTICHE GENERALI

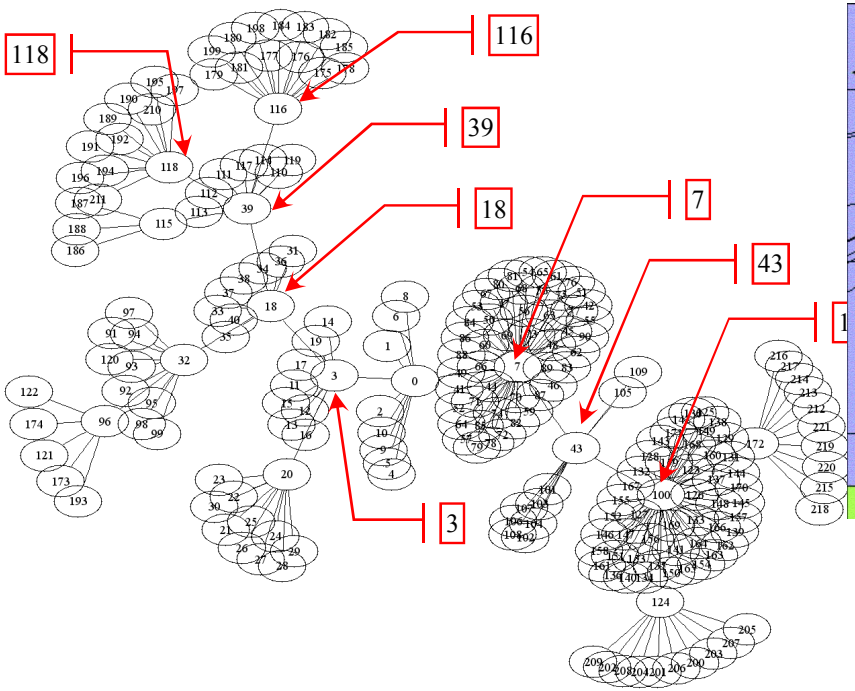
- Caratteristica principale, rispetto al P2P puro : definizione dinamica di un livello gerarchico nella rete
- Riduce il traffico relativo ai 'messaggi di controllo' senza ridurre l'affidabilità
- Superpeers (o UltraPeers): mantengono aperte molte connessioni con altri
- Leafnodes: mantengono aperte solo un numero limitato di connessioni.Tutte le connessioni sono verso SuperPeers (di solito uno)

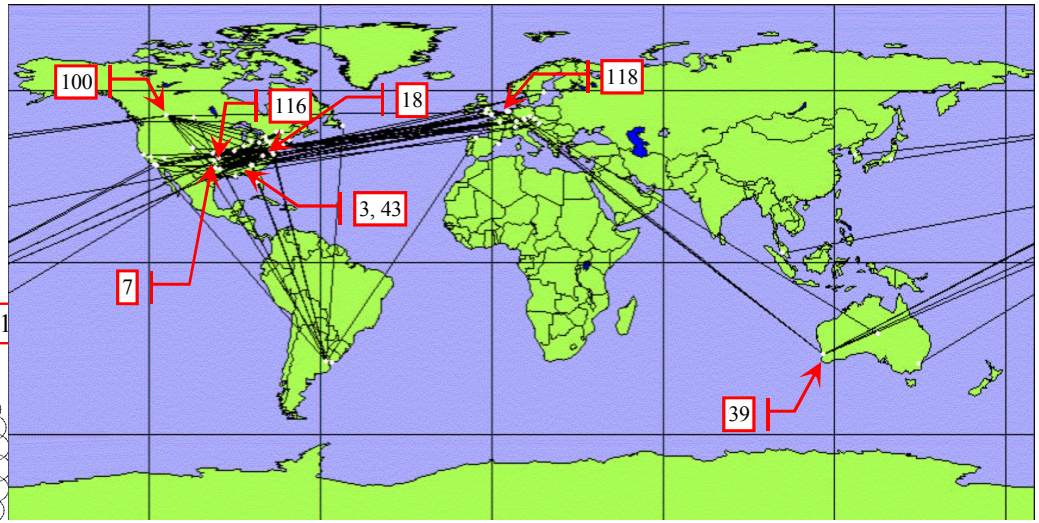
SuperPeer = Proxy verso la rete per i leafnodes gestiti

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. Osservare il cammino dal nodo 118 al nodo 18.
- La struttura dei Superpeer (hub strucure) è chiaramente rilevabile dalla rete astratta

GNUTELLA 0.6

- Obiettivi:
  - Metodo decentralizzato di ricerca di files
  - Diminuzione dei messaggi scambiati nelle reti P2P pure
  - Affidabilità paragonabile a quella di Gnutella 0.4 (non esiste un unico punto di centralizzazione)

- Alla base di diverse altre implementazioni di sistemi P2P (Kazaa,....)

- Breve Storia:
  - Primavera 2001: sviluppata a partire da Gnutella 0.4

  - Da allora:
    - Disponibili diverse implemetazioni (Limewire, Bearshare,...)
    - Definite diverse ottimizzazioni (privacy, scalability, performance,...)

GNUTELLA 0.6: TIPI DI NODI

L'architettura prevede due tipi di nodi

- Superpeer (UltraPeer):
  - 10-100 connessioni con i nodi foglia, <10 connessioni con altri ultrapeer
  - Proxy per i rispettivi nodi foglia
  - deve possedere certe 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 SUPERPEERS

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

SUPERPEERS: ELEZIONE DI SUPERPEERS

Criteri per la determinazione dei nodi SuperPeer Capable

- assenza di firewall. Di solito la verifica di questa proprietà viene approssimata verificando se il nodo ha ricevuto connessioni in ingresso
- banda di ingresso/uscita. Il calcolo della banda si approssima calcolando il throughput di upload/download
- quantità di RAM disponibile. Un SuperPeer deve memorizzare un insieme di routing tables, per indicizzare i files condivisi dai leaf nodes gestiti.
- Sistema Operativo. Alcuni sistemi operativi gestiscono un alto numero di sockets im modo più efficiente rispetto ad altri
- potenza di calcolo (ciclo di clock della CPU). Un SuperPeer deve essere in frado di gestire un alto numero di connessioni.
- Uptime tempo medio in cui un SuperPeer rimane attivo sulla rete Gnutella. Euristica: l'uptime futuro è proporzionale a quello passato

SUPERPEERS:APERTURA DI CONNESSIONI

- Foglia F → SuperPeer S
  - F diventa una foglia di S
  - F rifiuta connessioni di tipo 0.4
  - F invia ad S una QRP routing table
- Foglia F → Foglia G gestita da S: F diventa foglia di S
- Foglia F → Foglia G: Gnutella 0.4
- SuperPeer S1 → Superpeer S2
  - 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

CONNESSIONE FOGLIA-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

SuperPeer 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

CONNESSIONE CON FOGLIA GESTITA DA SUPERPEER

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

- Il nodo B rifiuta la connessione da A e '"ridirige" A verso il proprio ultrapeer S
- A non invia ok a B
- A tenta di stabilire una connessione con S e si comporta come nel caso visto nel lucido precedente

CONNESSIONE CON NODO GNUTELLA 0.4

- Se un nodo non è riuscito a trovare un SuperPeer, passa ad eseguire il vecchio protocollo Gnutella 0.4

Nodo A

GNUTELLA CONNECT/0.6

X-Ultrapeer: False

Nodo B

GNUTELLA/0.6 200

X-Ultrapeer: False

GNUTELLA/0.6 200 OK

APERTURA DELLE CONNESSIONI

- L'host SuperPeer Capable A si connette al SuperPeer B.

  Se B gestisce pochi nodi, può indicare ad A che non c'è necessità di diventare un SuperPeer. A diventa un LeafNode di B.

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	

- Un leafNode F può richiedere successivamente (ma solo quando è rimasto collegato alla rete per un intervallo di tempo sufficientemente lungo) al proprio superpeer S se deve diventare un superPeer

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 TABLES:VERSIONE BASE

Indicizzazione delle risorse condivise

- Ogni risorsa è individuata da un insieme di parole chiave
- Query Routing Protocol Table = Vettore di 65536 bits,contiene informazioni sui file condivisi
- Hhash 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.
- La posizione di indice v del vettore viene settata ad 1.
- Esempio: L'host A condivide un singolo file application.exe, individuato da diverse parole chiave, ad esempio software,JAVA,...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
- Routing table = Bitvector. La routing Table non memorizza il valore delle parole chiave, solo un insieme di flags che indicano la presenza dei valori delle chiavi

QUERY ROUTING TABLES:VERSIONE BASE

- Bloom Filters = Struttura dati utilizzata per rappresentare in modo efficiente un insieme di oggetti. Si basa su tecniche probabilistiche.
- Bitvector = Bloom Filter dell'insieme di parole chaive
- Risparmio di spazio a spese della precisione: E' 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 all'hash dei valori di contenuti in Q sono 0, la risorsa ricercata non è 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

ROUTING DELLE QUERY: VERSIONE BASE

- La QRT viene compressa, suddiviso in blocchi ed inviato al SuperPeer
- Il vettore viene memorizzato dal SuperPeer ed utilizzato come Query Routing Table per decidere se una query deve essere propagata ad quel leaf node
- Un LeafNode invia una richiesta al proprio Superpeer
- Il Superpeer controlla nella propria tabella di routing se il file richiesto è offerto da uno dei propri Leafnodes.
- Quando il SuperPeer riceve una query, applica le stesse funzioni hash applicate dal leaf node ad ogni parola chiave della query
- Il SuperPeer invia la query ad un Leafnode L solo se L possiede una risorsa che soddisfa la query
- Il Superpeer inoltre invia la query ad altri superpeers (quelli a cui esso è connesso nella overlay network) per individuare altri hosts che condividono il file richiesto
- Ad ogni richiesta inoltrata tra i superpeer viene associato un opportuno valore di TTL, per limitare la diffusione della richiesta stessa sulla rete dei SupePeers.

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 SuperPeer
  - 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 HTTP connessioni.

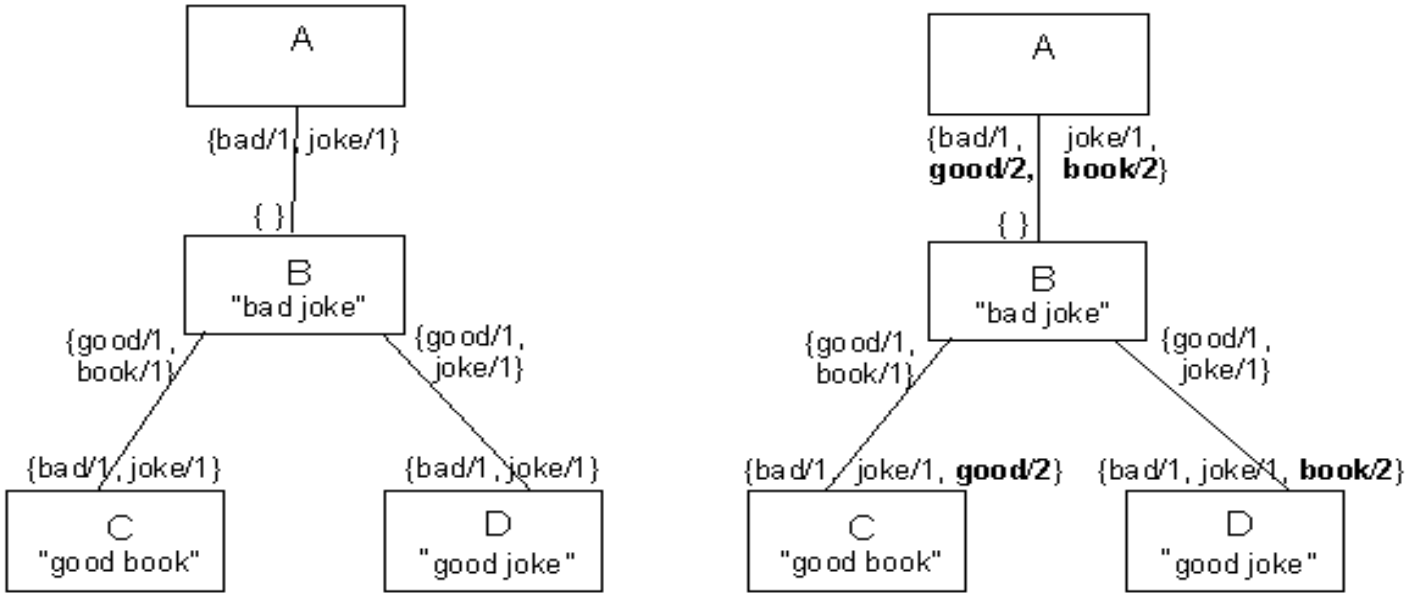
OTTIMIZZAZIONE DEL QUERY ROUTING

- 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

PROPAGAZIONE DELLE TABELLE DI ROUTING

- 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 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 è aggiornata nel modo seguente: per ogni posizione h della tabella di routing
  - se X 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 la ai propri vicini incrementando di 1 il valore di ogni posizione

PROPAGAZIONE DELLE TABELLE DI ROUTING



Propagazione delle tabelle di routing:

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

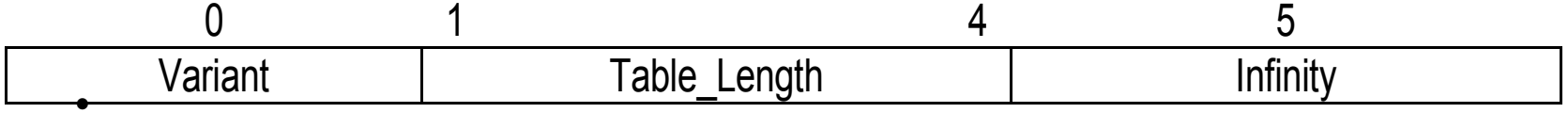
se hash('book')=1, hash('bad')=3,hash('good')=5,hash('joke')=7

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

GNUTELLA 0.6: QRP MESSAGES

Il QRP protocol utilizza il messaggio ROUTE_TABLE_UPDATE, con due diverse varianti

- RESET VARIANT : Utilizzato per segnalare (al superpeer o ad un superpeer vicino) che la tabella corrispondente al mittente deve essere resettata.



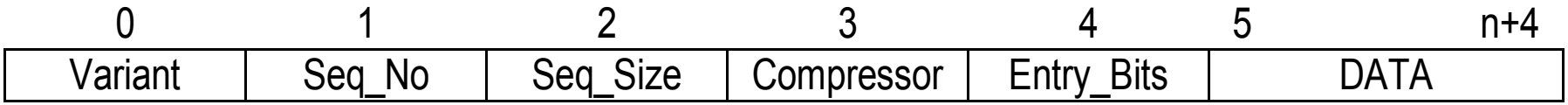
- Table_Length: Lunghezza della tabella
- Infinity:
  - Utilizzato per indicare ∞,
  - distanza massima verso un file = TTL+1

  Il ricevente alloca una tabella vuota di Table_Length posizioni e ne inizializza ogni componente con il valore indicato da INFINITY.

  Al messaggio RESET deve seguire una sequenza di messaggi di PATCH per comunicare i nuovi valori della tabella.

GNUTELLA 0.6: QRP MESSAGES

- ROUTE_TABLE_UPDATE, PATCH VARIANT: messaggio utilizzato per inviare aggiornamenti della tabella di routing



Compressor indica l'algoritmo utilizzato per la compressione dei dati

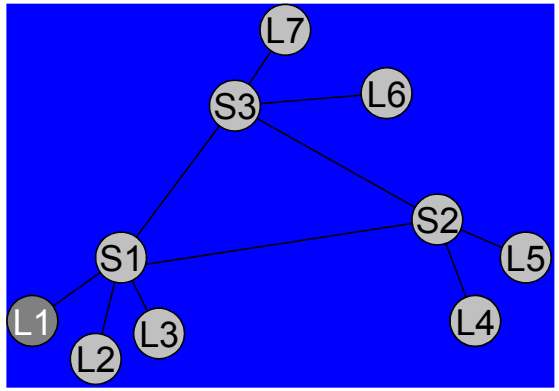
Seq_No, Seq_Size individuano il numero del messaggio

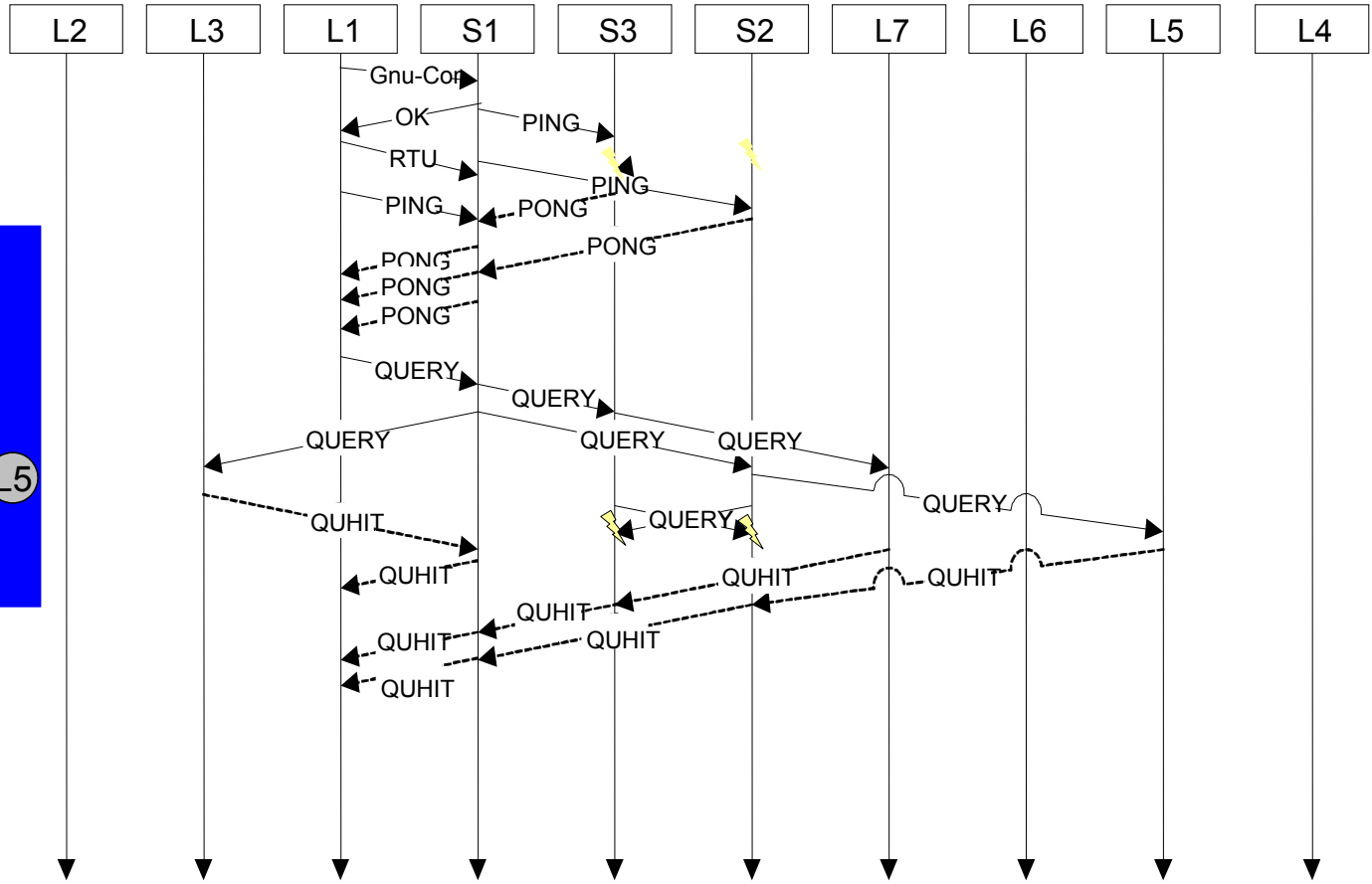
Data: Dati modificati

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: MESSAGGI DI CONTROLLO



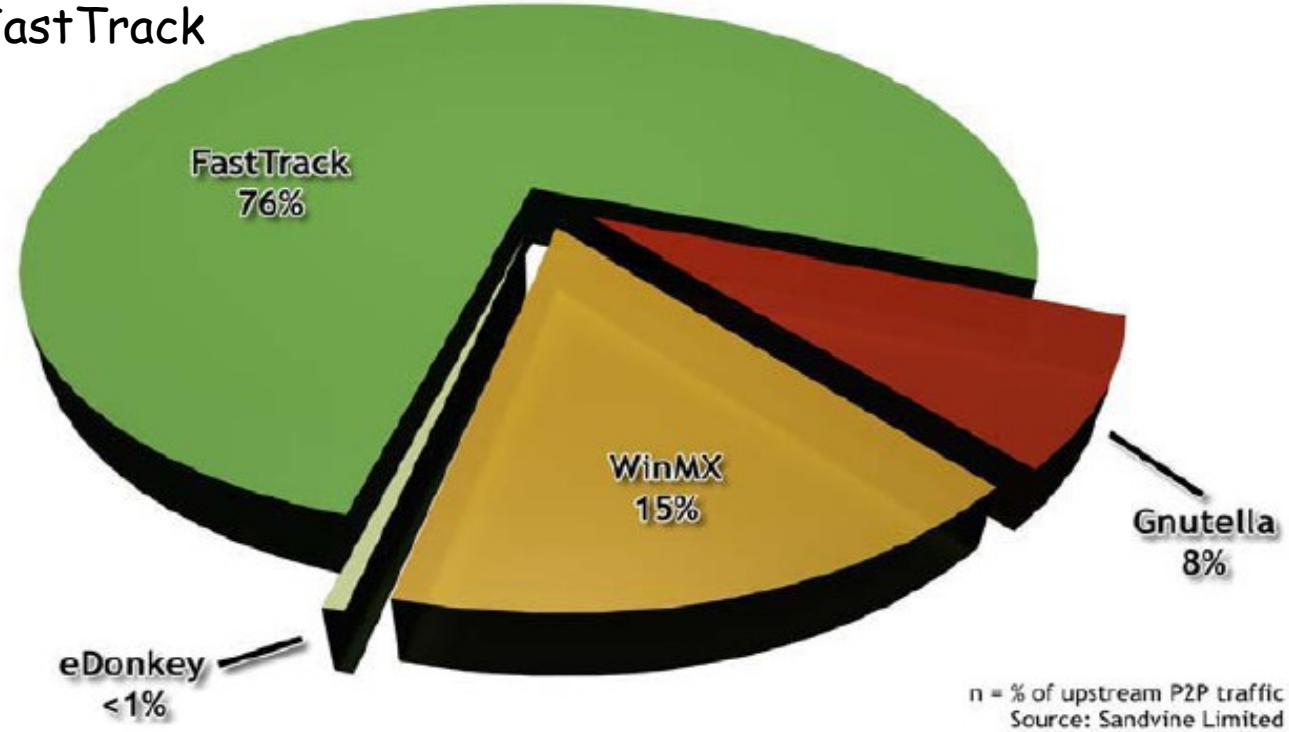


GNUTELLA 0.6: CONCLUSIONI

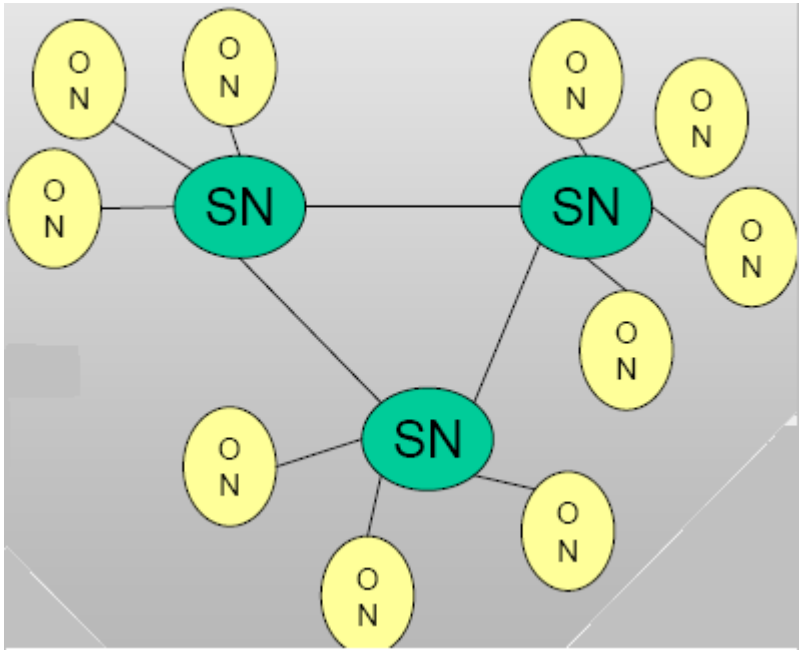
- I messaggi di ping pong vengono scambiati solamente tra i SuperPeers
- Un SuperPeer non propaga mai un messaggio di ping ai propri LeafNodes
- In questo modo ogni SuperPeer "mette al riparo" i propri LeafNodes dal traffico di keep-alive
- Un superPeer può ricevere un ping da un proprio LeafNode. In 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.

KAZAA: CARATTERISTICHE GENERALI

- Sistema P2P proprietario (2001), tutto il traffico è crittato (scarsa documentazione)
- Utilizza il protocollo FastTrack
- 76% del traffico P2P USA rilevato nel 2003 è dovuto a FastTrack



KAZAA: CARATTERISTICHE GENERALI



- Molte similitudini con 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
  - Ogni ON si connette ad un solo SN (connessione TCP)
  - I SN si connettono tra di loro e formano un'overlay network (connessioni TCP)

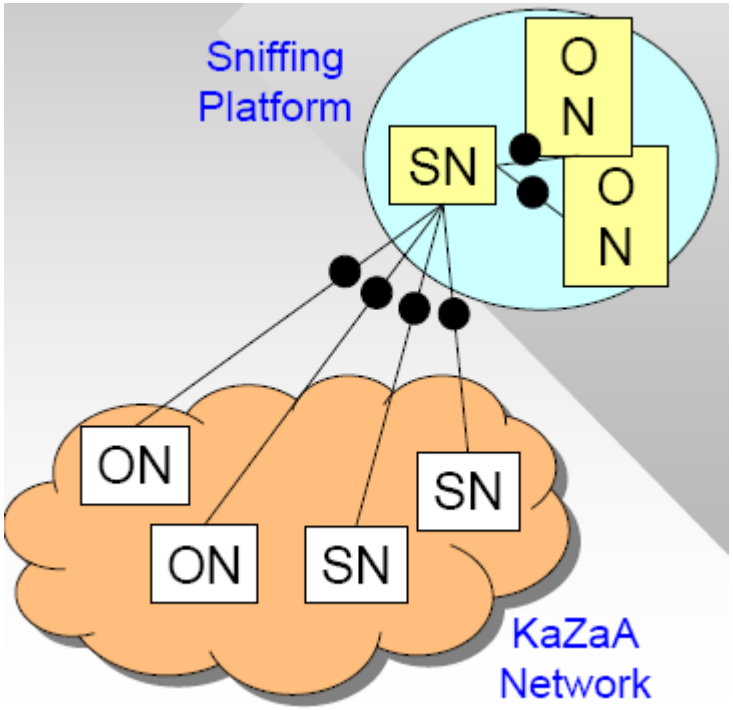
KAZAA: I SUPER NODES

- ON comunica al proprio SN i metadati che descrivono le informazioni che condivide (nome file, dimensione file, content hash del file, decrizione del file mediante parole chiave fornite dall'utente)

- 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'ovrelay

- Metadata = Nome, descrittore(autore, titolo,anno pubblicazione),dimensione del file, Content Hash (utilizzato per identificare il file univocamente)
- Content Hash = stringa di 20 bytes
  - i primi 16 bytes contengono l'MD5 dei primi 300k del file
  - ultimi 4 bytes sono l'hash della lunghezza del file

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
- Si analizza il traffico in entrata ed in uscita del SN

ANALISI SPERIMENTALE DELL'OVERLAY

Scopo dell'analisi dell'overlay Kazaa

- In media quanti nodi gestisce un Super Node? Quanti supernode esistono in un overlay Kazaa?
- Quale è la durata media di una connessione tra SN o tra ON e SN?
- Come avviene la scelta di un SN da parte di un ON?
- In Kazaa ogni SN ed ogni ON può selezionare la porta su cui attendere le connessioni. Come vengono gestite le porte? Che frazione di peer a monte di un NAT oppure di un firewall fa parte di una rete Kazaa?

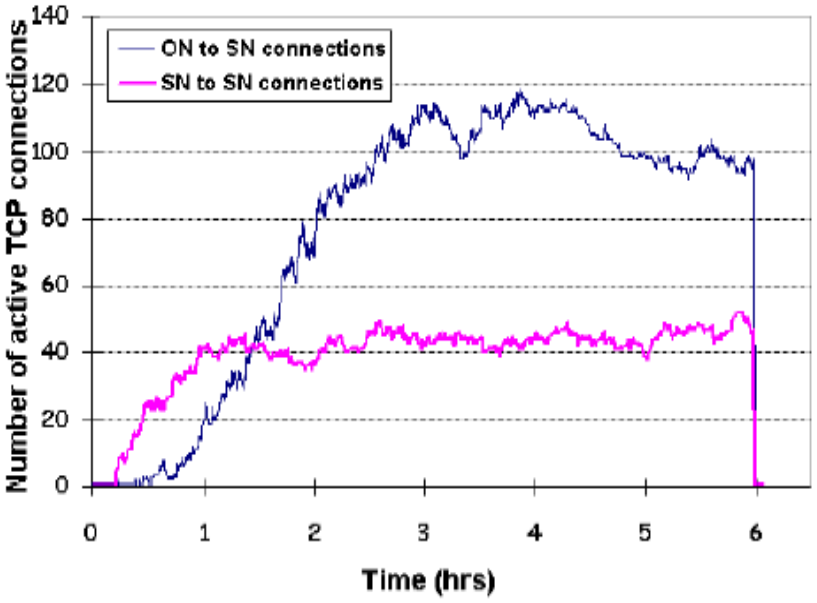
KAZAA: INGRESSO NELLA RETE

- Ogni nodo mantiene una cache di SN (SN cache list)
- Esistono default Server che possono essere interrogati se la cache è vuota
- SN Probing: All'inizio un ON prova ad inviare un pacchetto UDP a tutti gli 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
- Al momento della connessione ogni SN invia al proprio ON una lista contenente altri 200 SN. Queste informazioni sono utilizzate per popolare la SN cache list dell'ON

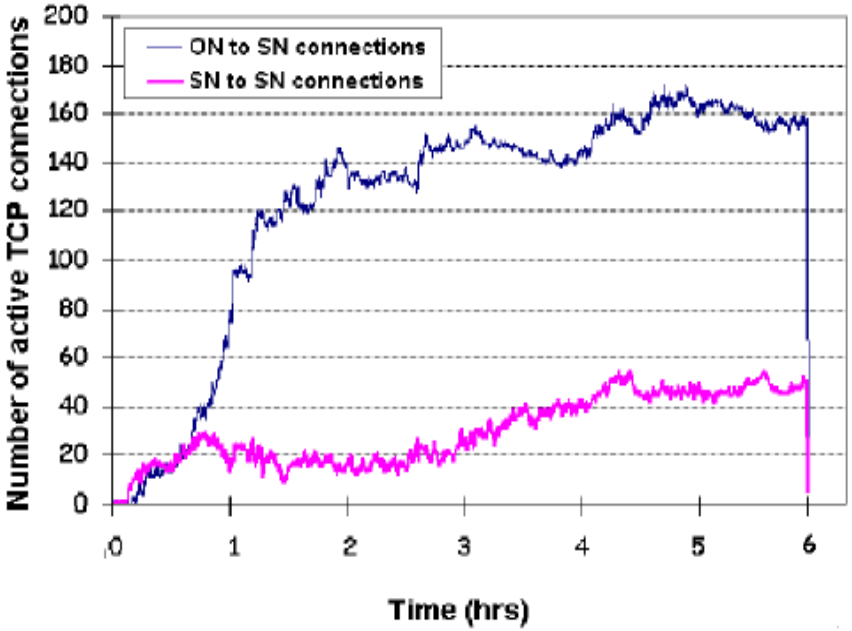
KAZAA: LOOK UP DEI DATI

- 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.
- SN inoltra la query ad un piccolo insieme di SN (algoritmo non noto).
- 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

CONNETTIVITA' DELL'OVERLAY



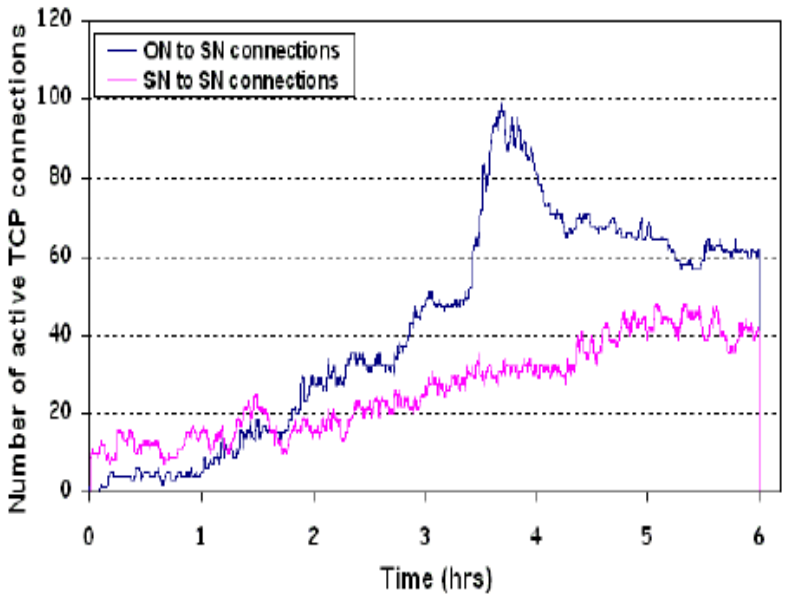
(a) Polytechnic campus - session evolution, Aug. 22, 2003



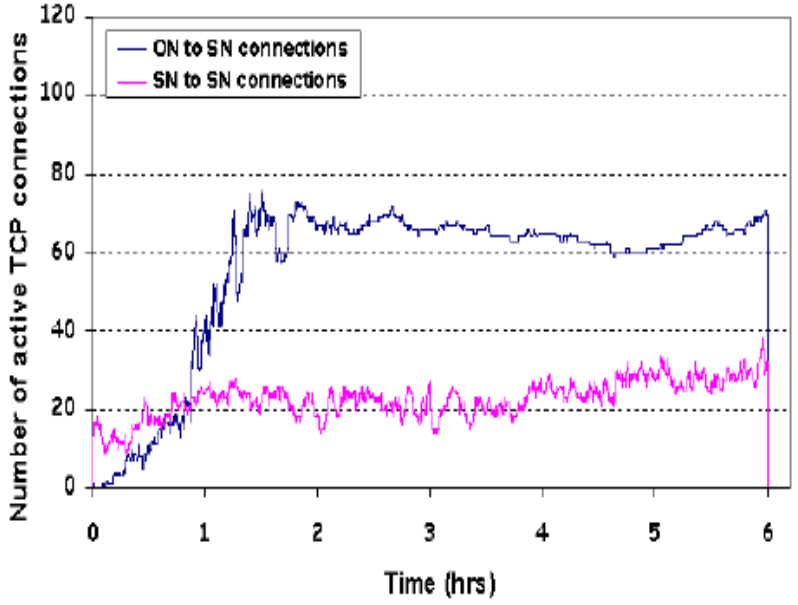
(b) Polytechnic campus - session evolution, Oct 24, 2003

- Analizza le connessioni tra il SN sniffer (installato al politecnico) ed altri ON e SN

CONNETTIVITA' DELL'OVERLAY



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



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

- Analizza le connessioni tra il SN sniffer(installato in una abitazione) ed altri ON e SN
- Diminuisce il nunero di connessioni con gli ON, mentre quelle tra I SN rimane simile

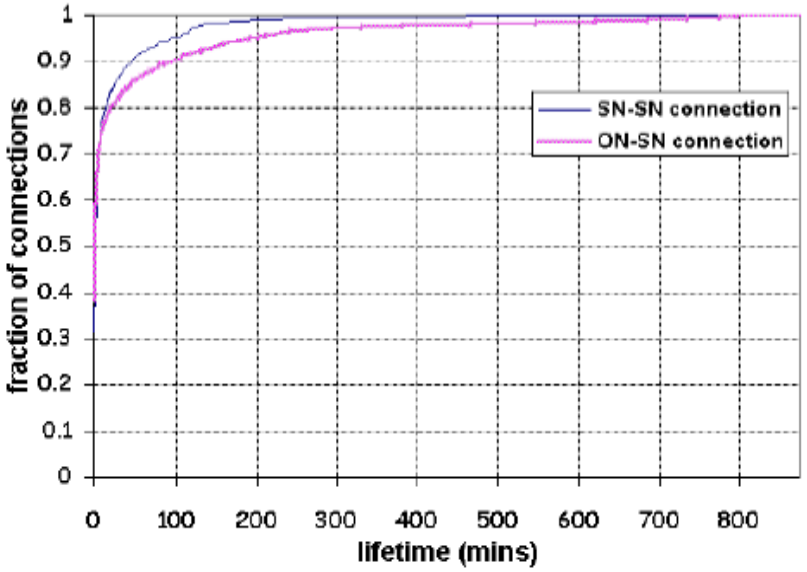
CONNETTIVITA' DELL'OVERLAY

- Il numero di connessioni tra il SN sniffer e gli altri peer (PN 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
- L'overlay tra i SN presenta pochi links

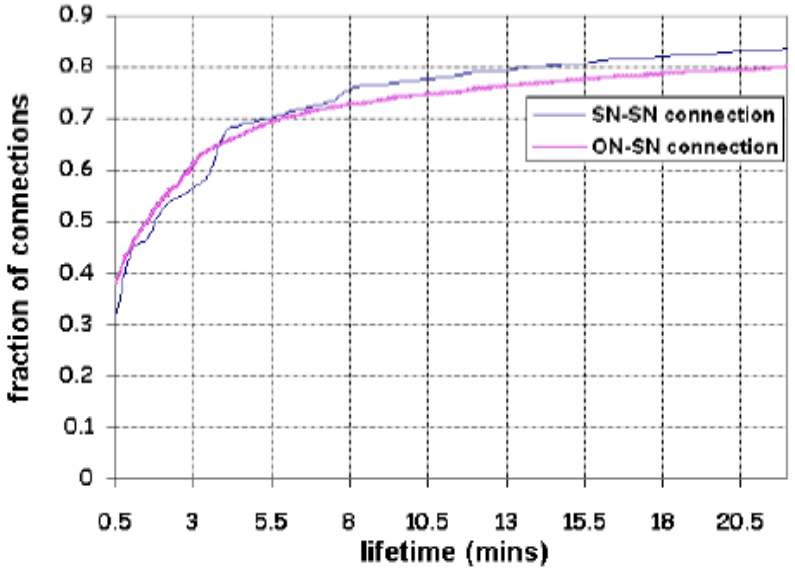
VALUTAZIONE DINAMICITA' DELL'OVERLAY

- Lo scopo del secondo insieme di esperimenti è di valutare la dinamicità dell'overlay
- Osservazione delle connessioni stabilite nell'arco di una settimana tra il SN sniffer e gli altri SN/ON.
- Risultato:
  - 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..... vedere slide successiva

VALUTAZIONE DINAMICITA' DELL'OVERLAY



(a) full duration plot



(b) close-up of the plot

- 38% delle connessioni tra gli ON e gli SN hanno una durata di meno di 30 secondi
- 32% delle connessioni tra SN ed SN hanno una durata di meno di 30 secondi
- Perche'???

VALUTAZIONE DINAMICITA' DELL'OVERLAY SN-SN

- L'overlay tra i SN è altamente dinamica. Ogni SN cambia le sue connessioni con altri SN molto frequentemente

- Motivazioni:
  - spesso un SN si connette ad un altro SN con il solo scopo di scambiare con esso la lista dei SN conosciuti, poi chiude la connessione.
  - Un SN tende a ricercare SN che presentano un carico basso
  - 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 leproprie possibilità di individuare le informazioni ricercate

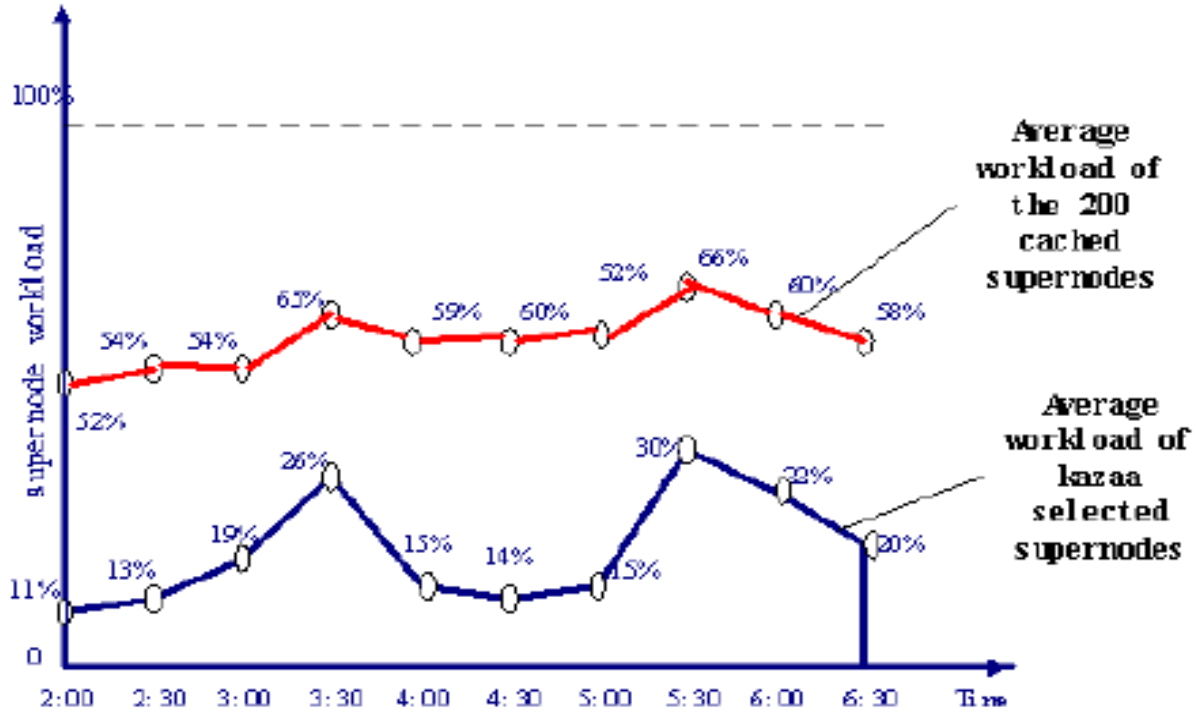
VALUTAZIONE DINAMICITA' CONNESSIONI ON-SN

- Anche le connessioni ON-SN variano frequentemente
- Motivazioni
  - Protocollo Kazaa: allo start-up un ON invia dei pacchetti UDP ai SN nella sua lista. Pi apre un insieme di connessioni con i SN che habbo 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
  - Un ON rinvia i propri metadati al nuovo SN ed il vecchio SN cancella i dati relativi all'ON disconnesso

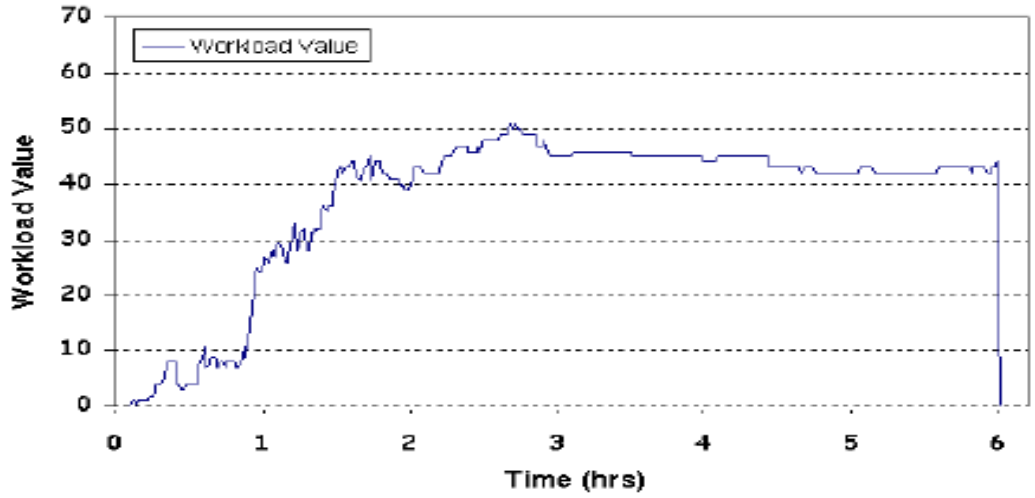
SELEZIONE DEI SN

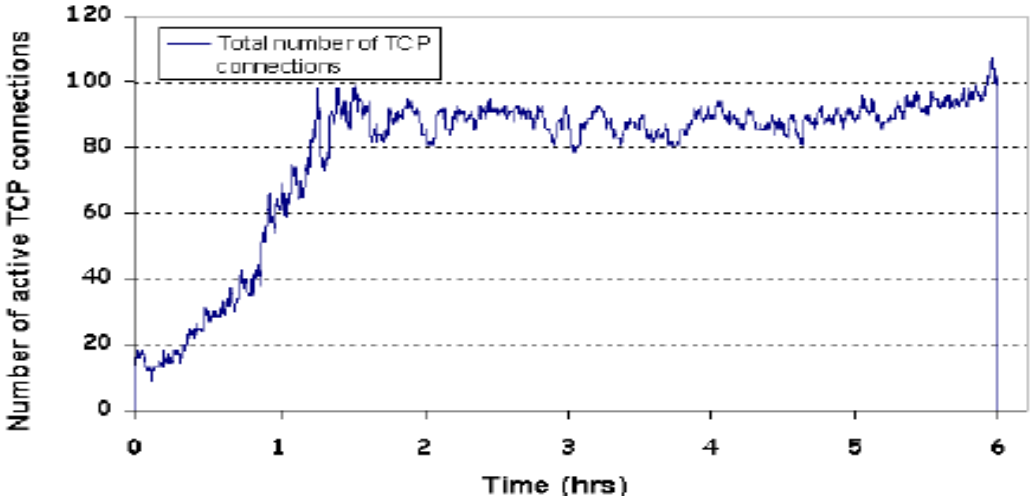
- Le connessioni ON-SN ed SN-SN vengono stabilite in base ai seguenti criteri:
  - Workload dei SN
  - Località
- Workload
  - Ogni ON mantiene una cache in cui registra una lista di SN
  - Per ogni SN vengono registrati: indirizzo IP, porta e workload (determinato prevalentemente in base al numero di connessioni gestite)
  - Supernode Workload: Un ON/SN sceglie preferibilmente un SN caratterizzato da un basso workload
  - La valutazione viene effettuata forzando un ON a connettersi a SN scelti dalla sua cache di SN e calcolando il rapporto del workload dei SN scelti rispetto a quello di quelli scartati

SELEZIONE DI SN



CARATTERIZZAZIONE DEL WORKLOAD





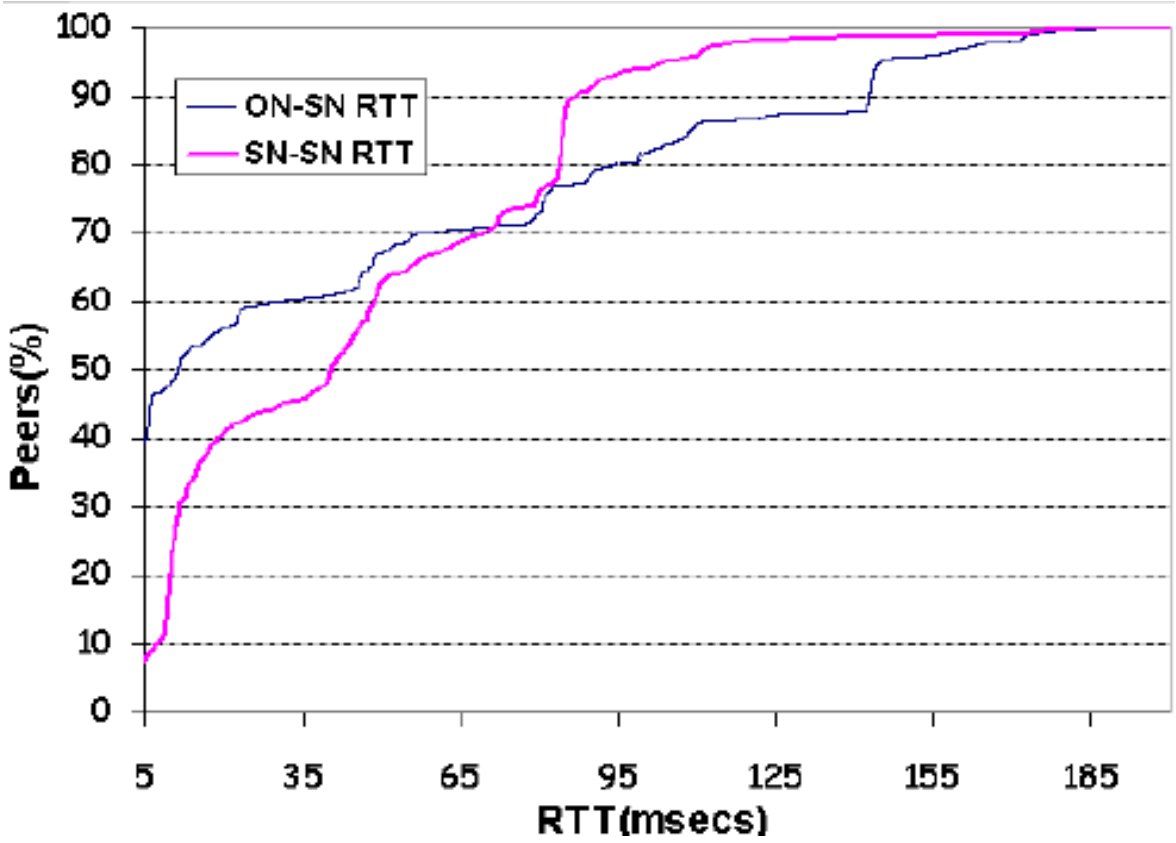
Esiste una forte correlazione tra

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

SELEZIONE DEI VICINI: LOCALITA'

- Località: Vengono scelti nodi che sono 'vicini', secondo qualche nozione di vicinanza
  - RTT Round Trip Time
    - 60% delle connessioni SN-SN presentano un RTT < 50 ms
    - 40% delle connessioni ON-SN presentano un RTT < 5 ms
    - Basso rispetto a valori tipici del RTT (100 ms tra l'Europa ed Nord America-Costa Occidentale, 180 ms tra America ed Asia)
  - Similarità rispetto ai prefissi IP

SELEZIONE DEI VICINI: RTT



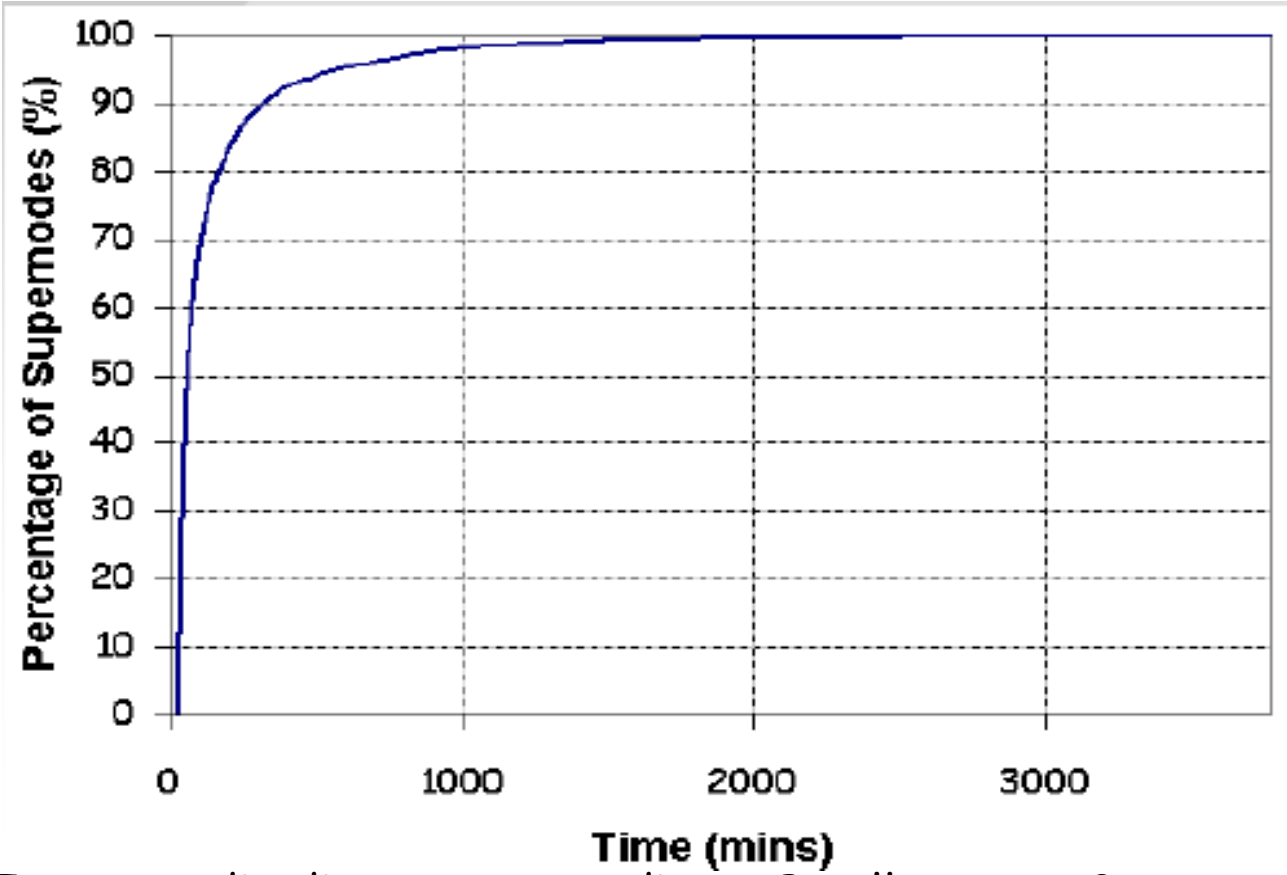
- Sull'asse delle Y percentuale di peer vicini (SN ed ON) la cui connessione presenta un RTT minore del valore riportato sull'asse delle x

DELEZIONE DEI VICINI:PREFIX MATCHING

Prefix Matching = Similarità rispetto agli indirizzi IP:

- Un ON si connette ad un SN
- Il SN 'padre' restituisce all'ON una lista L di SN
- L'esperimento consiste nel valutare la correlazione tra gli indirizzi IP degli SN in L e l'indirizzo IP di ON
- Risultato dell'esperimento: un alto numero di SN in L hanno prefissi IP 'simili' a quello di ON

PERMANENZA MEDIA DI UN SP SULL'OVERLAY



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

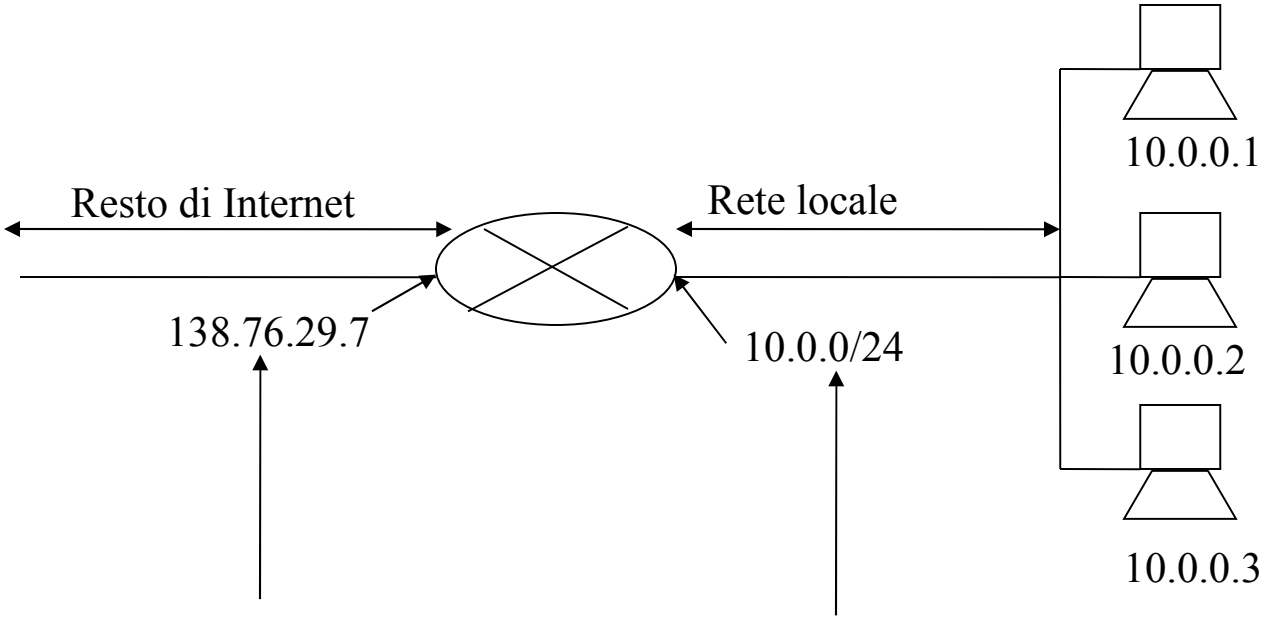
NAT E FIREWALL TRASVERSAL

- 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, onforma il SN sulla porta che sta utilizzando
- Questo meccanismo rende più complessa l'individuazione del traffico Kazaa

NAT (NETWORK ADDRESS TRANSLATION)

- NAT(Network Address Translation) = Tecnica di filtraggio di pacchetti IP con sostituzione degli indirizzi (mascheramento)
- Consente la connessione di un insieme di hosts ad Internet utilizzando un unico indirizzo IP
- Vantaggi
  - Risparmiare indirizzi IP (in attesa della piena diffusione di Ipv6). L'ISP attribuisce un solo indirizzo ad un insieme di hosts appartenenti alla stessa organizzazione. L'utente risparmia sul costo della connessione
  - facilita l'amministazione della rete:
    - si possono modificare gli indirizzi nella rete locale senza notificarlo al mondo esterno
    - Si può cambiare ISP senza modificare gli indirizzi locali
  - aumenta la sicurezza: gli hosts nella rete locale non sono visibili dall'esterno e quindi indirizzabili direttamente dall'esterno

NAT:NETWORK ADDRESS TRANSLATION



Tutti i datagrammi che escono dalla rete hanno lo stesso indirizzo NAT 138.76.29.7, ma diversi numeri di porta

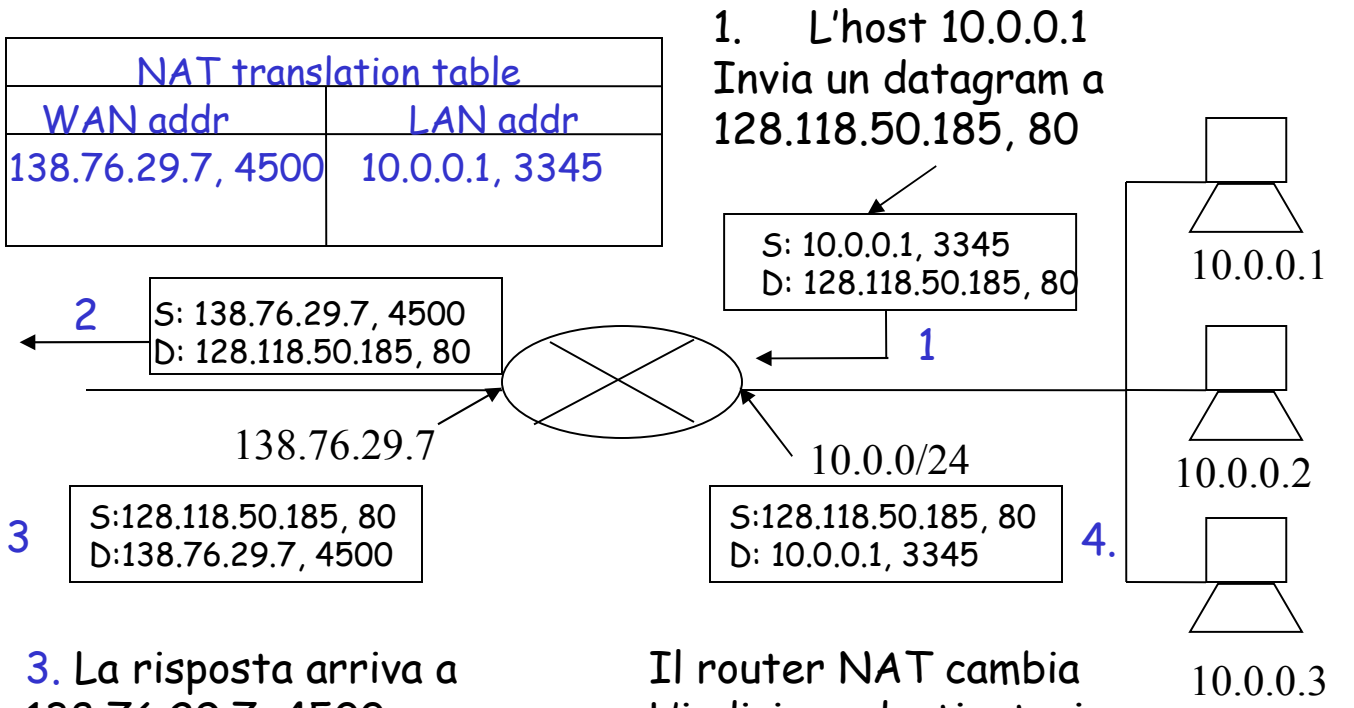
Tutti i datagrammi con sorgente/destinazione in questa sottorete hanno indirizzo 10.0.0/24 come sorgente/destinazione

NETWORK ADDRESS TRANSLATION

Funzioni di un router NAT

- sostituire in ogni datagram uscente la coppia (IP sorgente, #porta) con (NAT IP, #nuovaporta) dove #nuovaporta è un nuovo numero di porta generato dal NAT
- registrare in una tabella di traduzione la corrispondenza tra le due coppie
- Sostituire in ogni datagramma entrante la coppia (NAT IP, #nuovaporta) con il corrispondente (IP sorgente, #porta) memorizzato nella tabella di traduzione

NAT:NETWORK ADDRESS TRANSLATION



NETWORK ADDRESS TRANSLATION

- NAT : funziona solo con datagram IP che trasportano pacchetti a livello trasporto spediti mediante il protocollo UDP o il protocollo TCP
- Gli indirizzi assegnati alle sottoreti interne appartengono ad una delle seguenti zone
  - 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16
- Il router opera anche come dispositivo NAT
- Circa 60000 porte con 16 bit

  $\Rightarrow$

  circa 60000 connessioni aperte con un unico indirizzo IP

NETWORK ADDRESS TRANSLATION

- Svantaggi del NAT
  Il router manipola i numeri di porta (livello trasporto), mentre dovrebbe operare solo fine al livello 3 (IP)

- Rende più complessa la raggiungibilità degli host sulla rete ⇒
  difficoltà di sviluppo di applicazioni P2P

- Alcune applicazioni non sono trasparenti al NAT (esempio applicazioni che contengono indirizzi IP e numeri di porta nel payload)
  Esempio:FTP utilizza due connessioni parallele, una per l'interazione con il server, l'altra per il trasferimento dati da e verso il server.I parametri della seconda (porta su cui spedire i dati) connessione sono inclusi ne payload della prima

- Obiettivo: eliminare NAT con la diffusione di IPv6

NAT E FIREWALL TRASVERSAL

Connection Reversal:

- il peer A è nattato, ma ha già aperto una connessione TCP con il peer B
- C vuole aprire una comunicazione verso A
- C notifica a B la sua intenzione di comunicare con A
- B avverte A di aprire una connessione con C, utilizzando la connessione TCP già definita con A
- B si comporta come rely peer

Nel caso di Kazaa, il rely peer è il SN con cui l'ON nattato ha stabilito una comunicazione

CONCLUSIONI

- Kazaa può essere considerata una delle applicazioni P2P di maggior successo
- Riprende molte idea da Gnutella 0.6
- Le 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'
  - 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