Lezione n.6
Sistemi P2P: Gnutella
Laura Ricci

Materiale didattico:
Peer-to-Peer Systems
and Applications
Capitolo 5

Gnutella: il protocollo in breve

- Messaggi scambiati sulla Overlay Network
  - Messaggi di keep-alive: utilizzati per segnalare la presenza di un peer sulla rete
    - Ping: non contengono payload
    - Pong: risposta ad un ping, contengono informazioni sul peer contattato
  - Messaggi di richiesta di files condivisi
    - Query: contiene una stringa che include un insieme di parole chiave utilizzate per la ricerca
    - QueryHit: risposta positiva ad unmessaggio di Query
  - Push : utilizzato per consentire ad un peer a monte di un firewall di condividere i propri files.

- I messaggi di query hit e di ping vengono inviati mediante backward routing (seguono a ritroso lo stesso cammino dei rispettivi messaggi di query / ping).

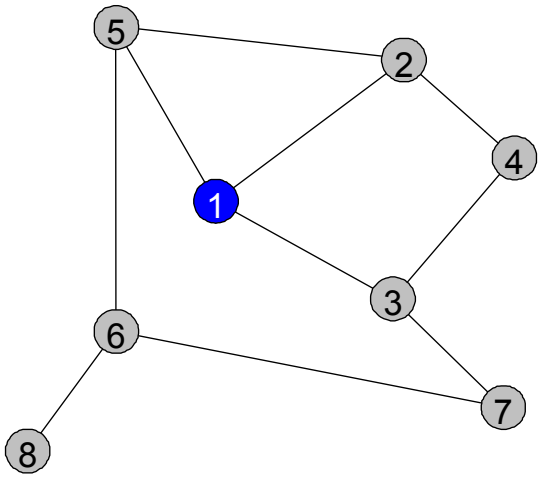
- Backward Routing: individuato mediante gli identificatori unici associati ai messaggi

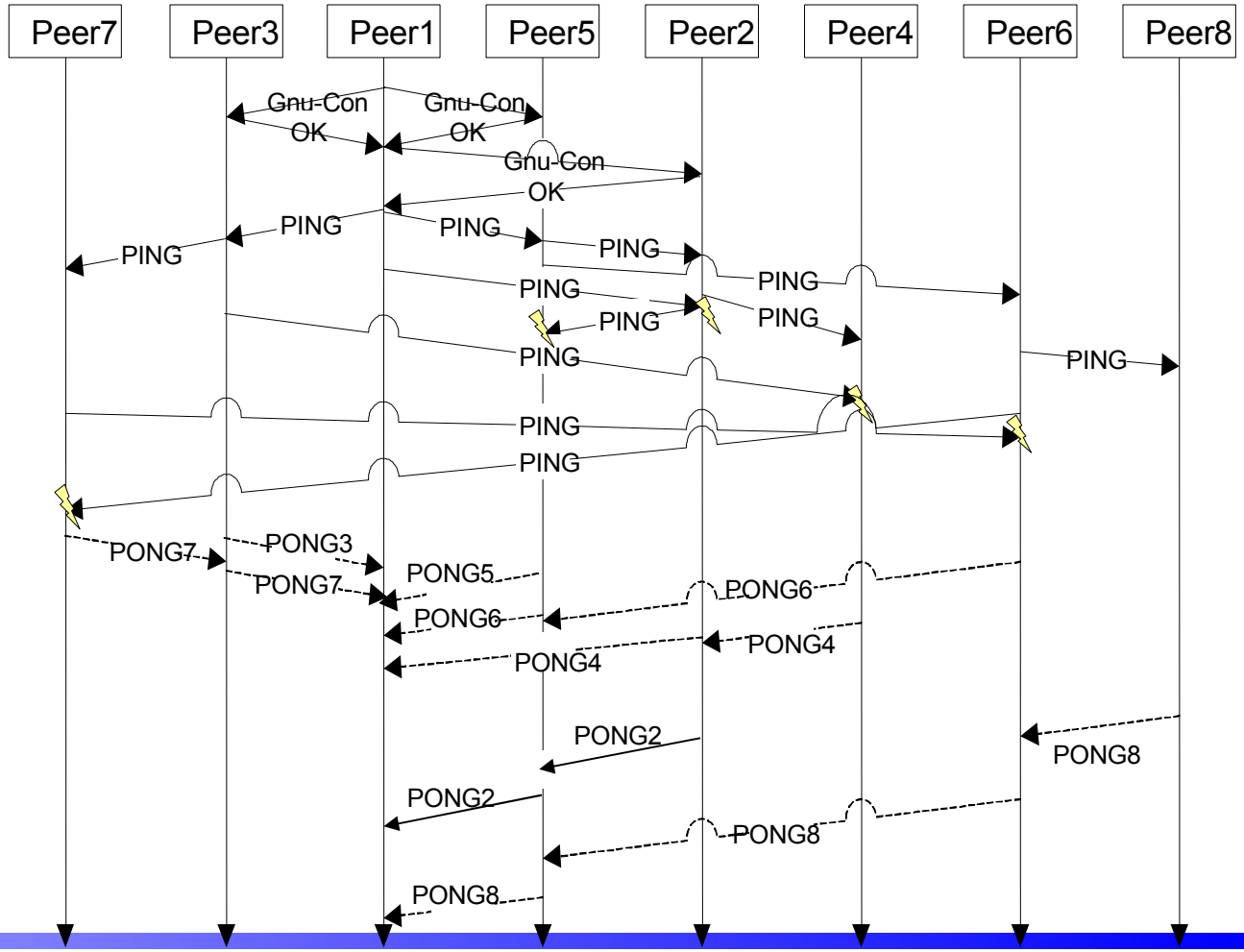
Gnutella:Bootstrap

Sequenza di messaggi scambiati nella fase di connessione ed esplorazione della rete

Numero di messsaggi per esplorare la rete:

- 6 connessione
- 12 PING
- 12 PONG





Gnutella: il Protocollo in Breve

- Perché i messaggi di query hit e di pong vengono inviati mediante backward rouring, invece che mediante una connessione diretta con l'host H che li ha originati?

  H dovrebbe accettare una gran quantità di connessioni

  Queste connessioni sarebbero utilizzata solo per inviare un numero limitato di dati

  Le connessioni verrebbero aperte ed immediatamente disatticate, dopo la ricezione di un pong o di un query hit

  Il backward routing consente di implementare politiche di caching (pong caching).

Gnutella: Analisi dettagliata del Protocollo

- In ogni sistema completamente decentralizzato è necessario risolvere il problema dell'individuazione di un punto iniziale di connessione:
  dove e come trovare un servent Gnutella a cui connettersi?

- Il meccanismo di bootstrap consente di stabilire la prima connessione ad un host della overlay network

- Un semplice meccanismo:

  invio di messaggi di ping ad un insieme di hosts scelti in modo completamente casuale sulla rete, senza accertarsi preventivamente se fanno parte della rete Gnutella

  Meccanismo inutilizzabile in quanto

  produce una grossa mole di traffico inutile

  il tempo di individuazione di un servent è in generale inaccettabile

Gnutella: Meccanismi di Bootstrap

Meccanismi implementati nelle prime versioni del protocollo:

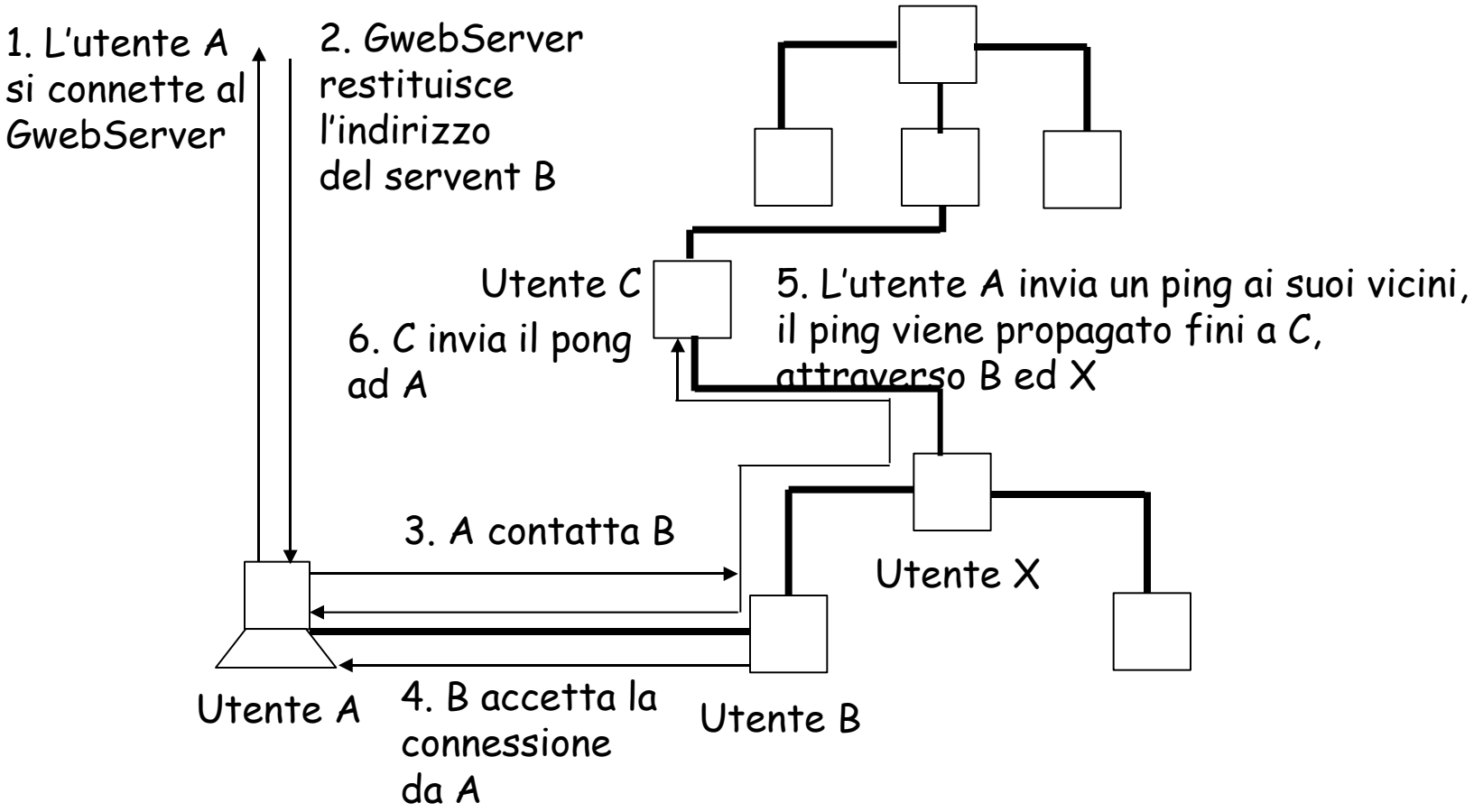
- Out of band methods: gli indirizzi di alcuni servent Gnutella sono pubblicati su alcune pagine web (o si possono ottenere mediante chat). L'utente passa uno di questi indirizzi al servent Gnutella, al momento della attivazione.

I meccanismi definiti nelle versioni successive sono basati su strategie di caching:

- GWebCache: sono dei web servers su cui è in esecuzione uno script che consente di interagire con servent Gnutella. Memorizzano gli IP di alcuni servent ed i riferimenti ad altri GWebCaches. E' aggiornato dinamicamente dai servents, in modo automatico.

- Cache interna al servent: memorizza gli indirizzi IP degli hosts con cui il servent è venuto a contatto durante la sessione attuale e le sessioni precedenti(ricavati da messaggi di pong, queryhit ed X-try)

Gnutella: Meccanismi di Bootstrap



Gnutella: GWebCaches

- Una ricerca di GWebCaches effettuata con Google restituisce migliaia di risultati (tuttavia molti sono relativi a GWebCaches non più attivi….).

- Definiscono un'ulteriore overlay network, definita "sopra" la rete Gnutella

- Ogni servent

  può richiedere al GWebServer l'indirizzo IP di un host oppure un riferimento ad altri GWebServers

  memorizza le informazioni ricevute dal GWebServer in una sua cache interna

  informa periodicamente il GWebServer della sua presenza nella rete

    gli aggiornamenti vengono inviati solo dopo che il servent è rimasto attivo sulla rete per un certo intervallo di tempo (esempio: un'ora). Dopo questo intervallo di tempo gli aggiornamenti vengono inviati periodicamente

    un servent non invia aggiornamenti al GWebServer se si trova a monte di un firewall

Gnutella: Local Caches

- Ogni servent Gnutella implementa un Local Cache. Il Local cache memorizza in modo permanente i riferimenti ad un centinaio di servent.
- La lista di servent viene caricata in memoria quando il servent viene lanciato e viene salvata al momento della sua disconnessione dalla rete.
- Una lista iniziale di hosts e GwebServers viene distribuita con il servent.
- La probabilità che uno degli hosts nella local cache sia attivo è alta, anche se il local host non è stato aggiornato da alcune settimane.
- Il local cache viene aggiornato dinamicamente con indirizzi di servent ricavati da interazioni con il GWebServer e dai messaggi di ping, di queryhit e di X-try (vedi fase di connessione)

Gnutella : Local Caches

- Alcune implementazioni di Gnutella mantengono nella RAM una lista molto grande di hosts contattati, ma al momento della disconnessione, salvano solo una parte della lista. Gli hosts sono scelti in base a
  - Uptime
  - Numero e dimensione dei files condivisi
  - Istante di tempo in cui il servent remoto è stato contattato
  - ……

- Il local cache deve essere utilizzato per evitare di contattare più volte durante la stessa sessione il GWebServer

- Il local cache viene utilizzato anche per evitare di ricontattare più volte lo stesso host

Gnutella : Un algoritmo di Bootstrapping

- Lanciare il servent e caricare i dati dal Local Cache
- Provare a connettersi alla rete Gnutella, utilizzando i dati ricavati dal Local Cache
- Se dopo X secondi non si sono stabilite connessioni, inviare una query al GWebServer
- Se dopo Y secondi non si è avuta risposta dal GWebServer, provare a contattare un altro GWebServer e così via, fino a che non si riesce a stabilire un contatto
- Non ricontattare il medesimo GWebServer durante la solita sessione
- Valori tipici: X= 5 secondi, Y= 10 secondi

Gnutella Handshaking

- Dopo che il servent A ha scelto un servent B a cui connettersi

  A stabilisce una connessione TCP con B

  A invia un messaggio a B costituito da una serie di righe. La struttura del messaggio è simile a quella di un messaggio HTTP

  - ogni linea è costituita da una sequenza di caratteri ASCII.
  - le linee sono separate dai caratteri <cr> <lf>
  - la prima linea è quella di richiesta/concessione connessione
  - Le linee successive sono righe di intestazione (headers)

  B può decidere di accettare la connessione o di rifiutarla (in base ad una politica di gestione delle connessioni)

Gnutella Handshaking

SERVLENT A

GNUTELLA CONNECT/0.6<cr><lf>
User Agent: Bearshare/1.0<cr><lf>
Ultra-Peer: False<cr><lf>
Pong-caching: 0.1<cr><lf><cr><lf>

GNUTELLA/0.6 200 OK<cr><lf>
<cr><lf>

SEVLENT B

GNUTELLA/0.6 200 OK <cr><lf>
User Agent:Limewire/1.0<cr><lf>
Pong Caching: 0.1 <cr><lf>
X-try:141.2.89.3:6436, 212.43.98.71:6436<cr><lf>

- X-try headers = Connection Pongs
- Contengono una lista di coppie (indirizzo IP, porta) che individuano una lista di servent Gnutella noti al servlent a cui ci si tenta di connettere. Vengono inviati anche se l'host decide di non accettare la connessione
- Esempio:
  X-try: 1.2.3.4: 1234, 3.4.5.6:3456

Gnutella: Gestione delle Connessioni

- Gestione delle Connessioni

Il numero di connessioni aperte dipende dalla banda di entrata/uscita che caratterizza il servent

Una semplice politica per gestire le connessioni:

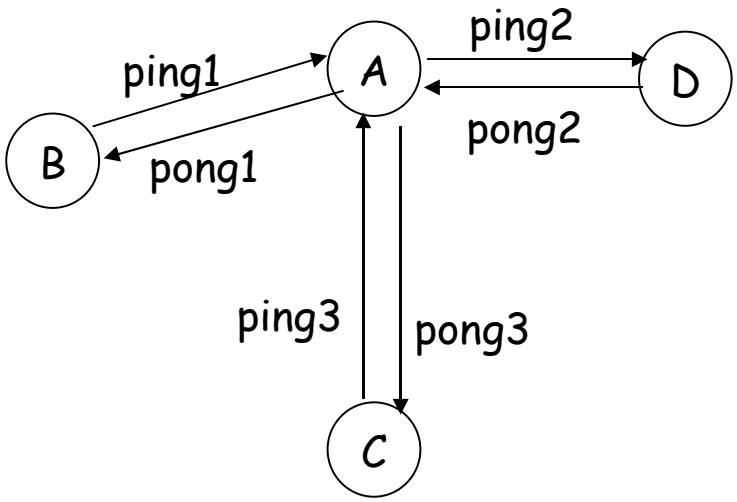
stabilire un valore k che indica il massimo numero di connessioni che il servent può accettare.

Durante la fase di bootstrap aprire n (n<k) connessioni verso altri servent.

I rimanenti n-k slots sono utilizzati per accettare connessioni in ingresso, durante la permanenza dell'host all'inteno della rete Gnutella.

Pong Caching

- Pong Caching: introdotto per diminuire il numero di messaggi sulla overlay network



Quando A riceve il messaggio ping3 utilizza per inviare pong3 l'informazione ricevuta dal precedente messaggio di ping (ping2)

Messaggi di Pong: Un semplice Algoritmo di Caching

- Memorizzare per ogni connessione un vettore di messaggi di pong
- Quando un servent riceve un messaggio di pong sovrascrive il messaggio di pong memorizzato meno di recente
- Per ogni pong memorizzato, si registra Indirizzo IP, Porta, numero di files e di kilobytes condivisi, ed il valore di Hops.
- Il valore di Hops indica quanto è distante sulla rete il servent che ha inviato il pong.
- La cache viene aggiornata periodicamente. Il servent invia un ping su tutte le connessioni ogni X secondi. Il valore di X varia a seconda che il vicino adotti o meno una strategia di caching dei messaggi di Pong (informazione ricevuta dal vicino al momento dell'apertura della connessione)

Messaggi di Pong: Un semplice Algoritmo di Caching

- Al momento della ricezione di un ping Pi, il servent invia un insieme di messaggi di pong, costruiti a partire dai messaggi presenti nella cache pong cache

- Scelta dei pongs:

  pongs provenienti da connessioni diverse

  Scegliere pongs con HOP counts diversi

- Per ogni messaggio di pong Po costruito a partire da un messaggio di pong Pc, individuato nella cache:

  L'identificatore di Po è uguale a quello di Pi

  Il valore H di HOPS di Po è ottenuto incrementando di 1 il valore HOPS di Pc

  Il valore TTL è assegnato in modo tale che TTL+H = 7

  Se il valore del TTL ottenuto è inferiore al valore HOPS di Pi, Po non viene spedito

Gnutella: Trasferimento dei files

- Quando un servent A riceve un messaggio di query hit da un servent B, A può decidere di scaricare il file richiesto da B . In questo caso apre una connessione direttamente con B.

- Il download

  non avviene tramite la overlay network,

  È implementato mediante protocollo HTTP, su una connessione diretta tra servents

- Se B si trova a monte di un firewall, può non essere in grado di accettare connessioni

  B avverte A di trovarsi a monte di un firewall (esiste un flag nel messaggio di query hit)

  Al momento della ricezione del query hit, A invia indietro a B un messaggio di push

  Quando B riceve il push, apre una connessione verso A, quindi invia il file.

Gnutella: Il controllo del flusso

- Controllo del flusso a livello applicazione: garantisce di non sovraccaricare le connessioni aperte
- Uso di code associate alle connessioni
- Controllare la dimensione delle code
- Stabilire una politica di scelta dei pacchetti da inviare nel caso di connessioni sovraccariche
- Una politica ragionevole:

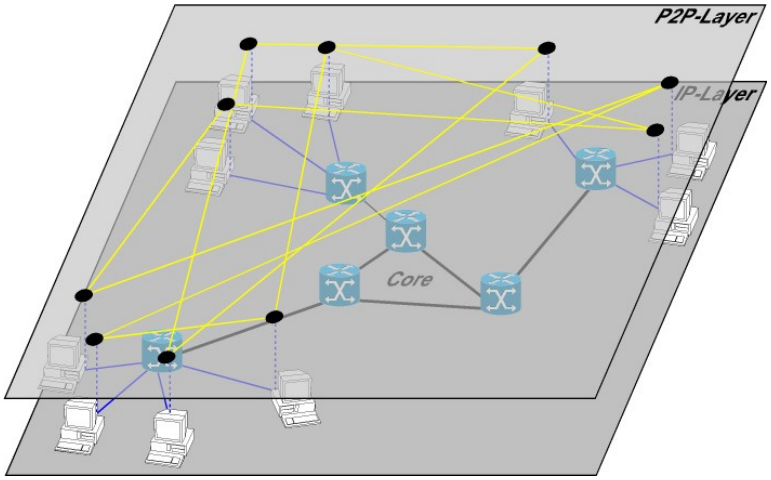
  Assegnare priorità diverse ai messaggi (es: Push, Query Hit, Pong, Query, Ping)

  Per i messaggi di Query e di Ping la priorità è assegnata in modo inversamente proporzionale al valore di hops del messaggio. Le queries locali hanno priorità massima

  Per i messaggi di Query-hits, Pong la priorità è assegnata in modo direttamente proporzionale al valore di hops del messaggio. Si evita così di sprecare inutilmente banda di comunicazione

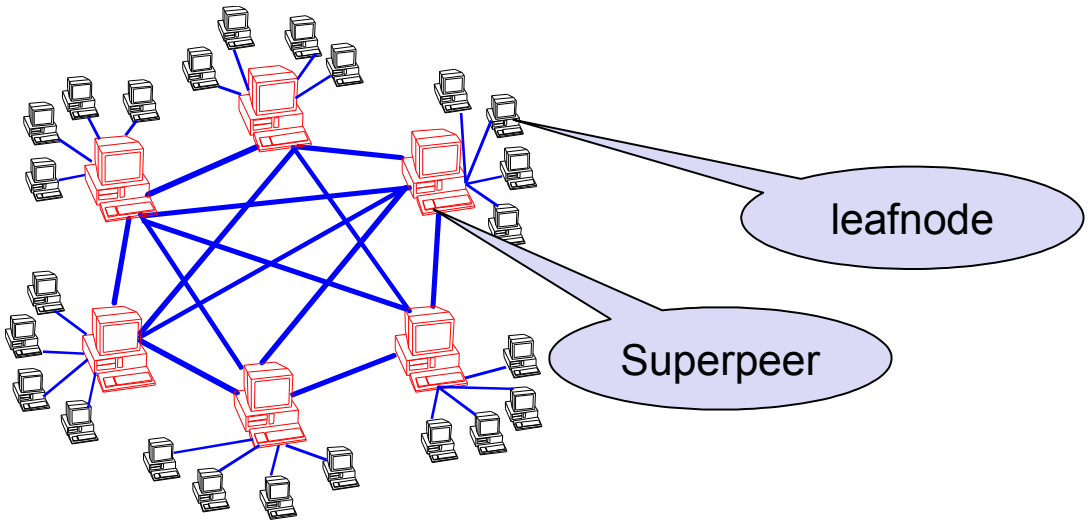
Riassunto della Presentazione

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

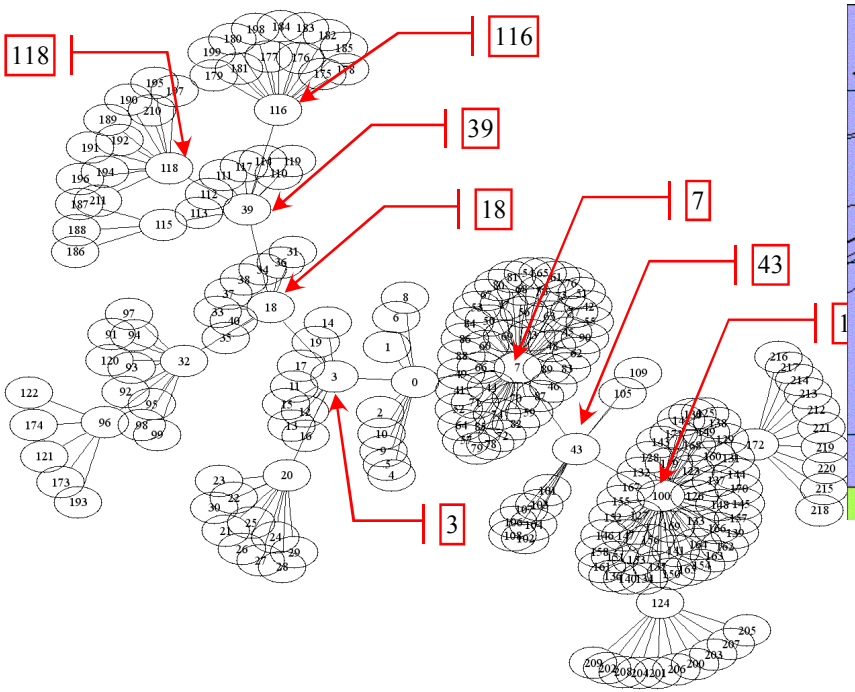


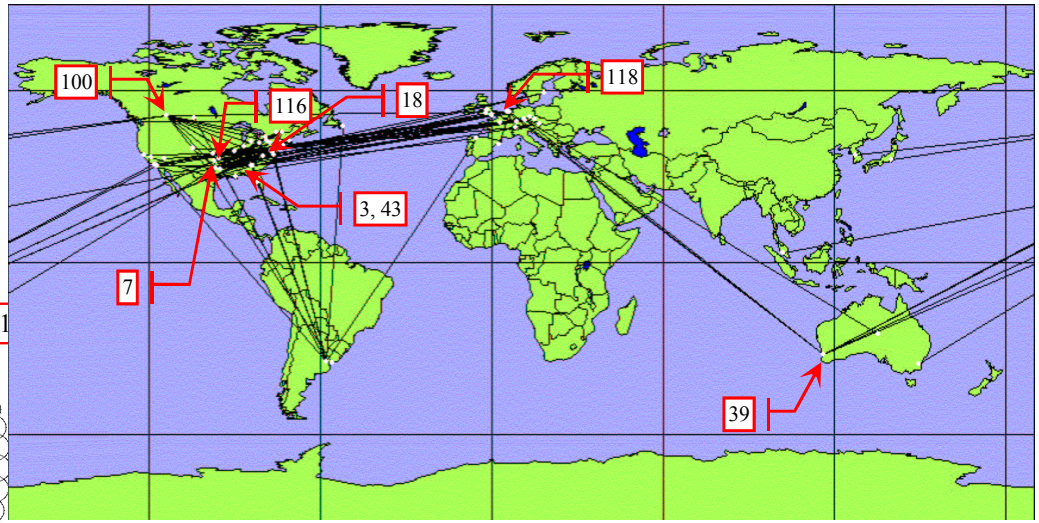
Definizione di P2P Ibrido

- 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, grado>>20, dipende dalla dimensione della rete
- Leafnodes: mantengono aperte solo un numero limitato di connessioni (grado<7). 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 messaggi scambiati nelle reti P2P pure
  - Stessa affidabilità (non esiste un punto di centralizzazione)

- Alla base di diverse altre implementazioni di sistemi P2P (non BitTorrent)

- Breve Storia:

  Primavera 2001: sviluppata a partire da Gnutella 0.4

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

Elezione Dinamica dei SuperPeers

- L'elezione dei SuperPeer avviene in modo completamente decentralizzato

- 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

- Un host H effettua il bootstrap alla rete Gnutella secondo la procedura definita in Gnutella 0.4 (GWebCache, local caching,…), modificata come segue

  Se H si qualifica come LeafNode, ricerca un SuperPeer

  Se H è SuperPeer Capable ricerca altri SuperPeer e controlla, interagendo con questi ultimi, se è necessario inserire nuovi SuperPeers sulla rete Gnutella. In caso affermativo, H diventa un SuperPeer, altrimenti unLeafNode

Elezione Dinamica dei SuperPeers

Criteri per la determinazione dei nodi SuperPeer Capable

- non deve trovarsi a monte di un firewall
- sufficiente banda di ingresso/uscita
- quantità di RAM disponibile. Un SuperPeer deve memorizzare un insieme di routing tables, per indicizzare i files condivisi dai leaf nodes gestiti.
- 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.

Elezione Dinamica dei SuperPeers

- Un LeafNode tenta di connettersi ad un SuperPeer

- Un SuperPeer tenta di connettersi ad altri SuperPeers

  Esempio il l'host SuperPeer Capable A si connette al SuperPeer B. Se B non gestisce un numero ragionevole di nodi, B indica ad A che non c'e' necessità di diventare un SuperPeer. A diventa un LeafNode di B.

  Il Protocollo di handshaking:

SuperPeer 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	

Gnutella 0.6: Notifica dei Files Condivisi

- Dopo la connessione ad un SuperPeer, un LeafNode notifica al proprio Super Peer tutti i files che intende condividere (content notification)

- Il SuperPeer costruisce un insieme di tabelle di routing

- Definizione di messaggi di ROUTE_TABLE_UPDATE

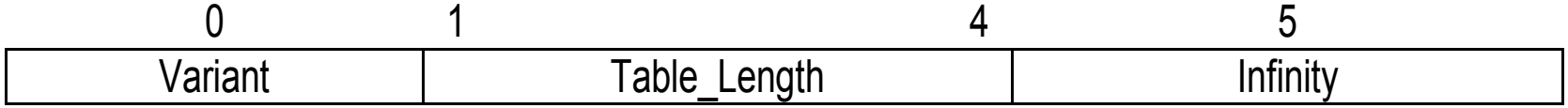
  ROUTE_TABLE_UPDATE (0x30), Reset variant (0x0): per eliminare una tabella di routing

  ROUTE_TABLE_UPDATE (0x30), Patch variant(0x1): per inviare al SuperPeer una nuova tabella di routing

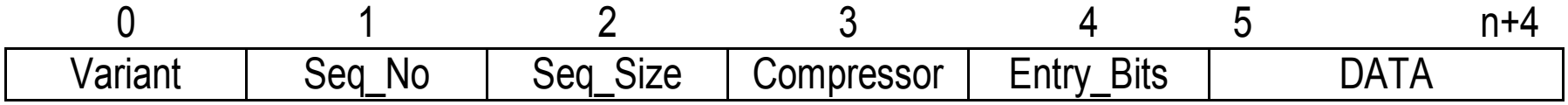
Gnutella 0.6: Formato dei Messaggi

- Ogni LeafNode possiede un elenco di parole chiave che individuano i files condivisi
- Ad ogni parola viene applicata una funzione hash e viene inserito un "flag di presenza" nella corrispondente posizione della hash table
- La hash table non memorizza le parole, solo un insieme di flags che indica la presenza di alcune chiavi
- Il vettore booleano viene compresso, viene suddiviso in blocchi e inviato al SuperPeer
- Il vettore viene memorizzato dal SuperPeer ed utilizzato per effettuare il routing delle queries

Gnutella 0.6: Formato dei Messaggi



Reset Variant: utilizzata per eliminare la tabella di routing di un leaf node da un SuperPeer



Patch Variant: utilizzato per inviare una parte della tabella di routing di un Leaf Node

Seq_No, Seq_Size individuano il frammento inviato e la sua lunghezza

Data: Vettore di booleani

Gnutella 0.6: Formato dei 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: il Routing delle Queries

- 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.
- Il SuperPeer invia la query ad un leafnode L solo se l'hash applicato ad almeno una delle parole chiave contenute nella query corrisponde ad un valore contenuto nella routing table corrispondente ad L.
- 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.

Gnutella 0.6: il Routing delle queries

- 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)

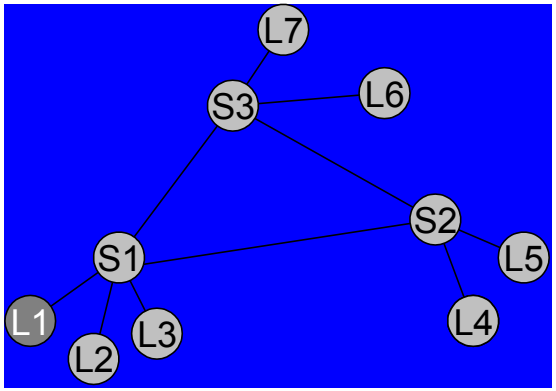
- Scambio dei files:

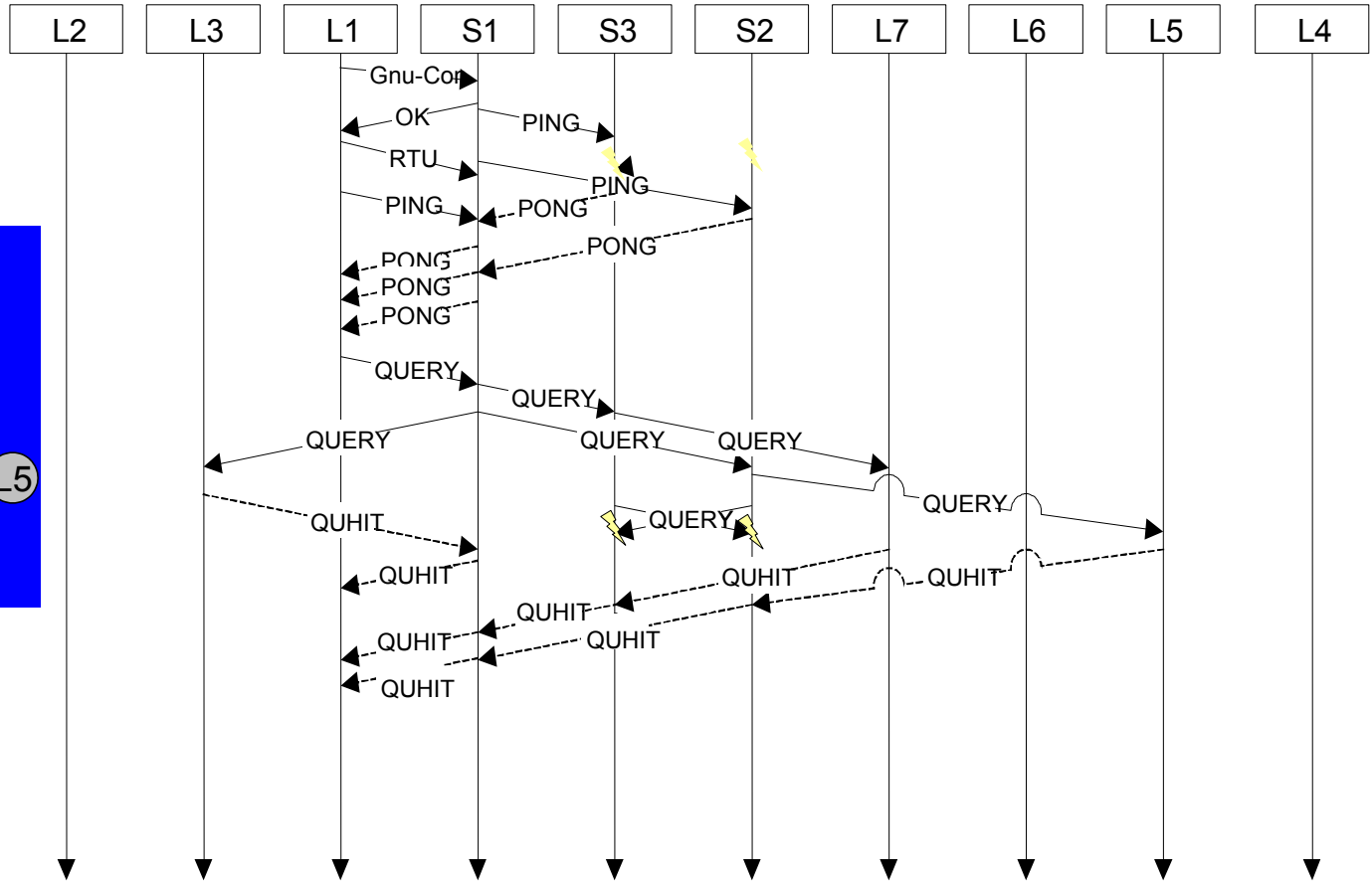
  Avviene direttamente tra i nodi interessati, tramite HTTP connessioni.

Gnutella 0.6: i messaggi di Keep-alive

- 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.

Gnutella 0.6: Messaggi di Controllo





Ulteriori Sistemi Basati su il Modello ibridi

- Edonkey
- Kazaa/FastTrack
- Emule
- OpenNap
- …