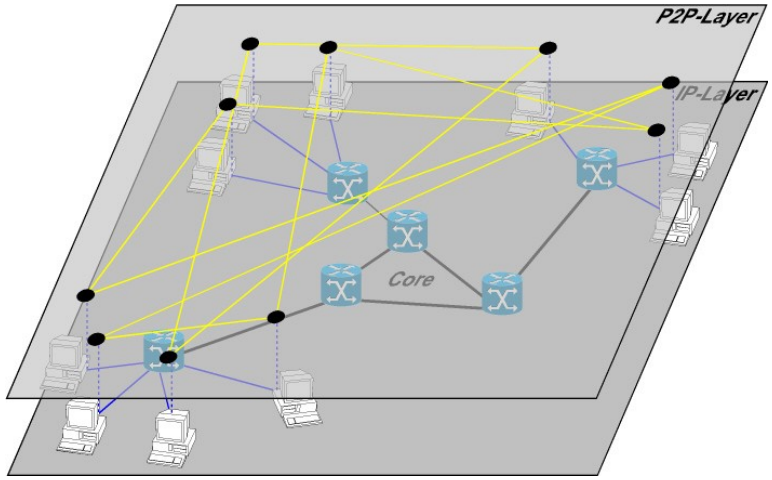
Lezione n.2
Gnutella 0.4

Laura Ricci

27/2/2009

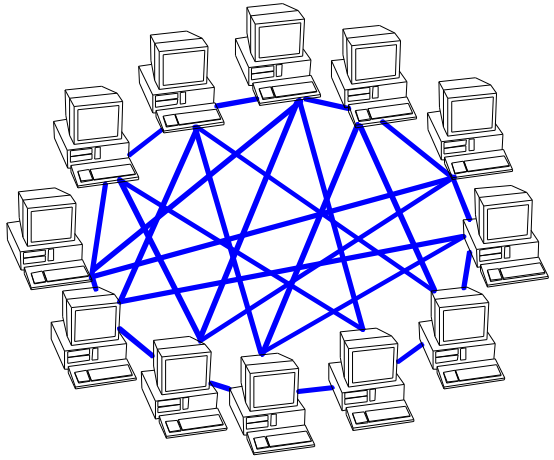
RIASSUNTO DELLA PRESENTAZIONE

1. Caratteristiche generali dei sistemi P2P di prima generazione
2. Reti P2P centralizzate
   - Caratteristiche Base
   - Protocollo
   - Dicussione
3. Reti Peer to Peer Pure
   - Caratteristiche Base
   - Protocollo
   - Discussione
4. Reti Peer to Peer Ibride
   1. Caratteristiche Base
   2. Protocollo
   3. Discussione



SISTEMI P2P PURI

- Non esiste una entità centralizzata nella overlay network
- La rimozione di un peer non riduce le funzionalità del sistema
- Le connessioni tra i peer sono stabilite per:
  - propagare le queries e le risposte alle queries
  - acquisire conoscenza sullo stato della rete



P2P PURO: CARATTERISTICHE GENERALI

- Non esiste un server di indirizzo noto con cui stabilire la connessione iniziale: è necessario definire una opportuna procedura di bootstrap.
- Soluzioni possibili:
  - Definizione di bootstrap servers (Beacon Server= Sever "Faro").
    Web Server che memorizza una lista di peer attivi (con alta probabilità) sulla rete
  - Utilizzo di peer-cache.
    Ogni peer mantiene in una propria cache una lista di peer contattati nelle sessioni precedenti

P2P PURO: CARATTERISTICHE GENERALI

- Routing delle queries: Completamente Decentralizzato

- Reactive protocol
  - i peer non notificano i files che intendono condividere
  - i cammini verso i peer che forniscono il file ricercato (content providers) vengono stabiliti dinamicamente (on demand)

- Invio richieste (queries): utilizza un meccanismo di enhanced flooding
  - limitato dal TTL
  - utilizzo di GUID (Global Identificator) per identificare univocamente i messaggi che transitano sulla rete

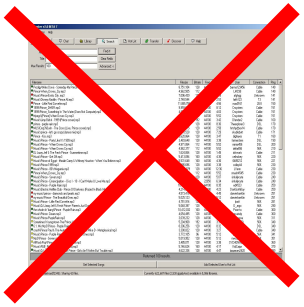
- Invio delle risposte: backward routing.
  - La risposta positiva ad una query viene inoltrata a ritroso lungo lo stesso cammino seguito dalla quesry
  - utilizzo del GUID per individuare il backward path

FLOODING

- Flooded Request Model:
  - Non esiste una autorità centralizzata di coordinamento (tutti i peer sono uguali)
  - La richiesta di ricerca viene inviata ad un numero di peer predeterminato
  - Se un peer non può soddisfare la richiesta, la passa ad altri peer, fino al raggiungimento di una determinata profondità di ricerca (ttl=time-to-live, centralizzata)
  - Quando il file richiesto è stato localizzato, il risultato positivo della ricerca viene rinviato al peer che lo ha richiesto
  - Il peer che ha richiesto il file, lo può scaricare direttamente dal peer che offre tale fiel
- Caratteristiche:
  - L'individuazione del documento non è garantita.
  - Ridotta scalabilità.
- Esempi: Gnutella0.4, FastTrack "Super Peers" come proxies.

FLOODING

http://www.gnutelliums.com/



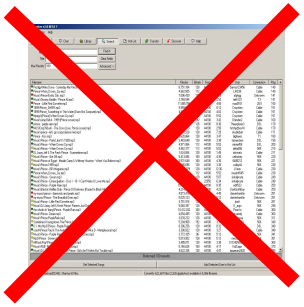
Non esiste un Database Centralizzato

X ricerca Prince

Y fornisce Prince

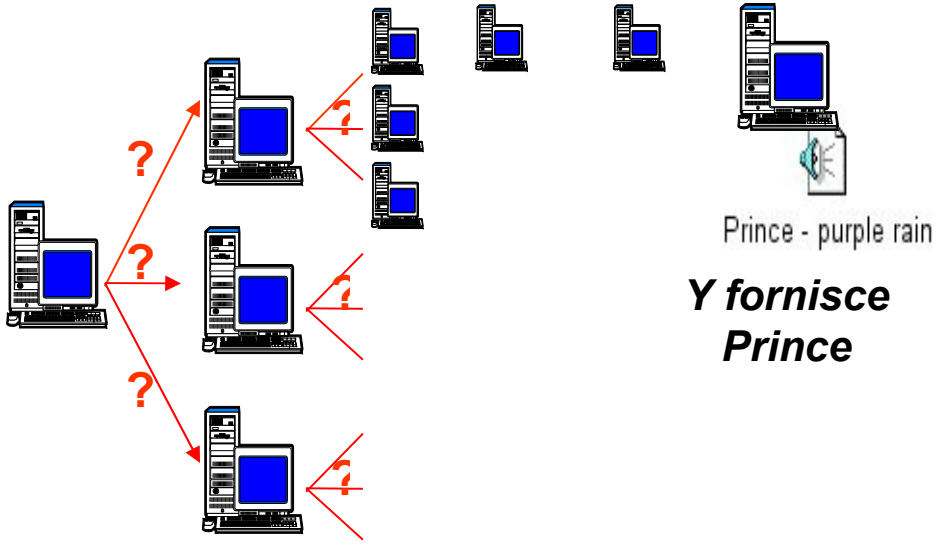
FLOODING

http://www.gnutelliums.com/



Non esiste un Database Centralizzato

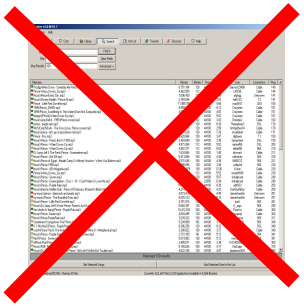
X ricerca Prince



Y fornisce Prince

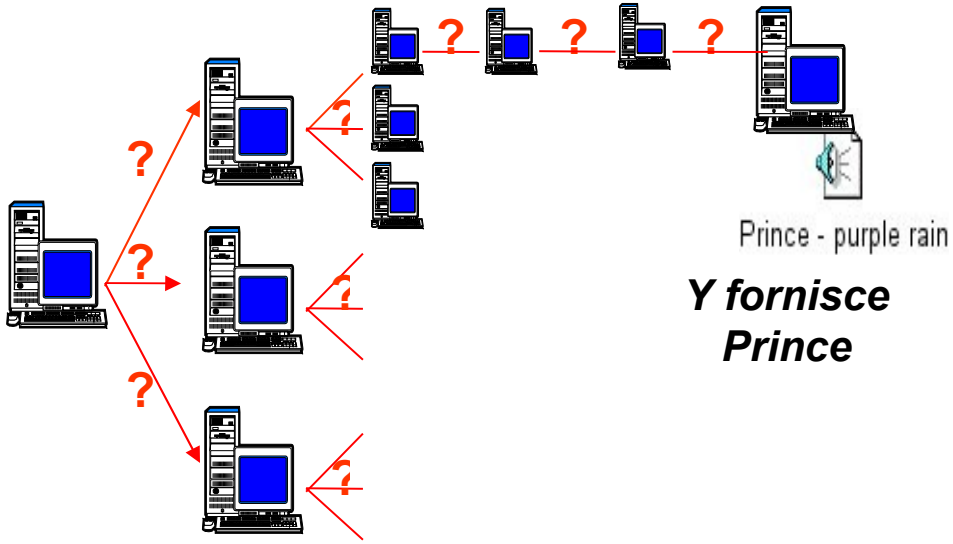
FLOODING

http://www.gnutelliums.com/



Non esiste un Database Centralizzato

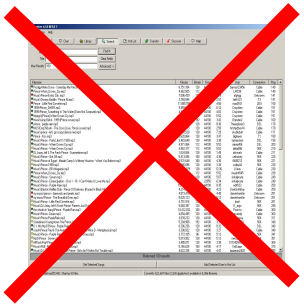
X ricerca Prince



Y fornisce Prince

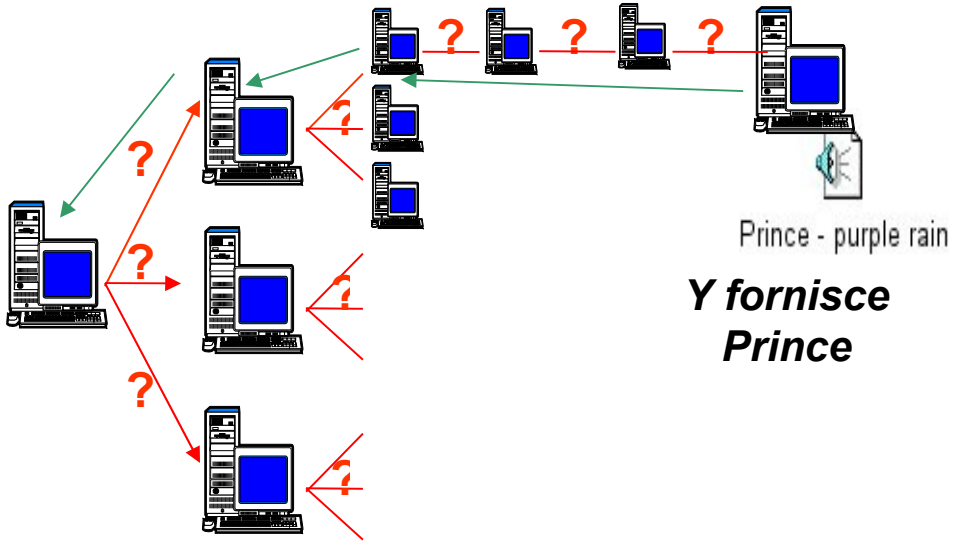
FLOODING

http://www.gnutelliums.com/



Non esiste un Database Centralizzato

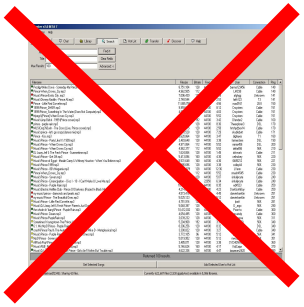
X ricerca Prince



Y fornisce Prince

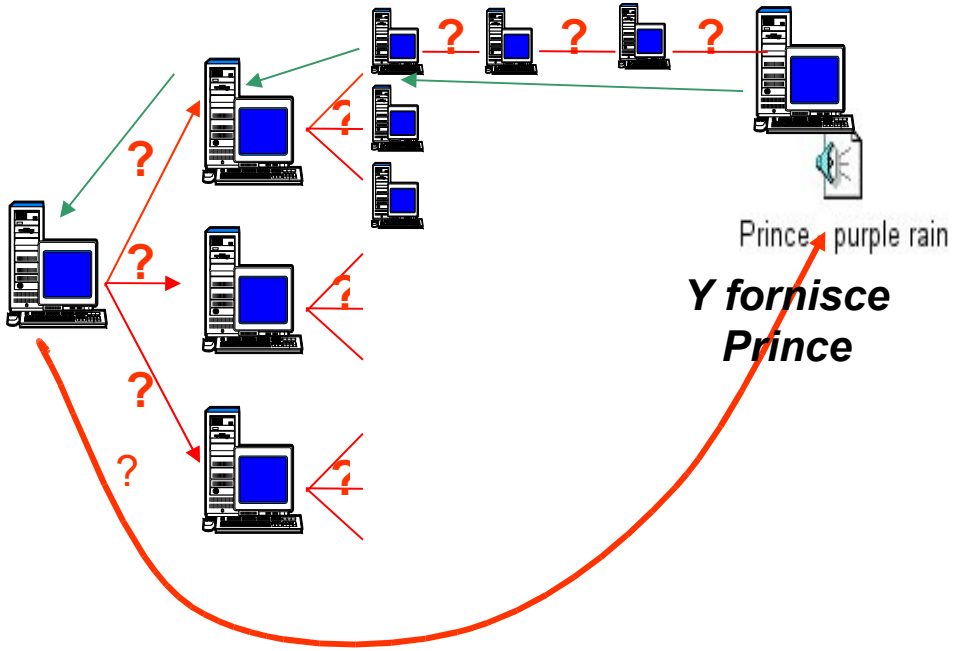
FLOODING

http://www.gnutelliums.com/



Non esiste un Database Centralizzato

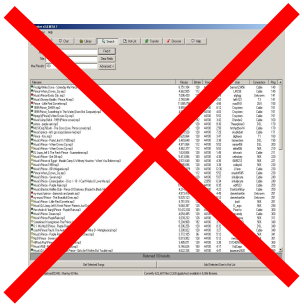
X ricerca Prince



Y fornisce Prince

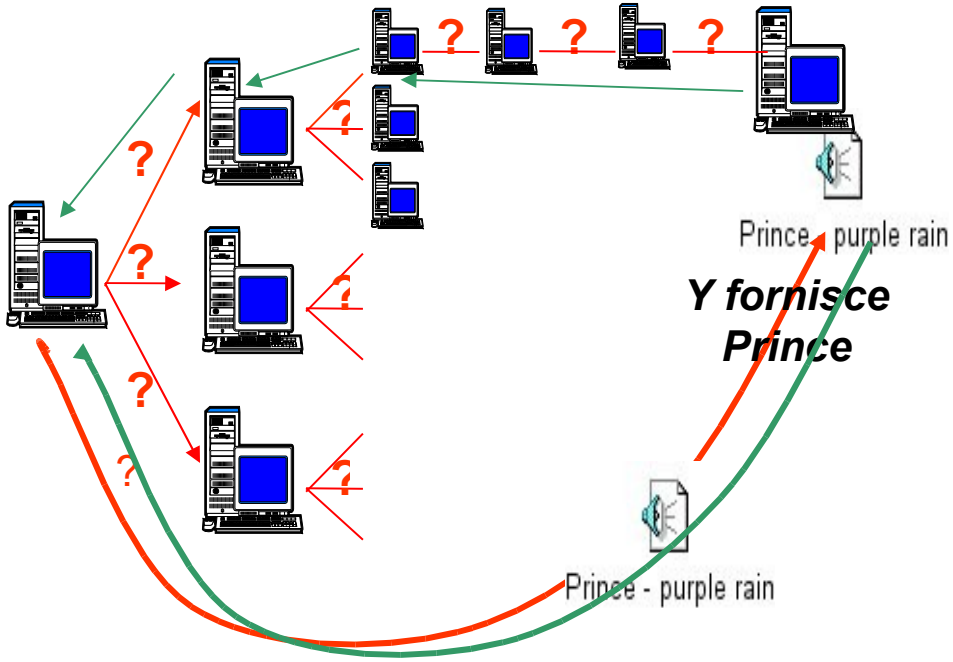
FLOODING

http://www.gnutelliums.com/



Non esiste un Database Centralizzato

X ricerca Prince



Y fornisce Prince

P2P PURO: CARATTERISTICHE GENERALI

Signaling Connections

- Definiscono la overlay network
- Utilizzate per
  - la ricerca di informazione condivisa
  - invio di messaggi di keep-alive
- Stabili, vengono modificate solo quando cambiano i vicini sulla overlay network
- Basate su TCP

Content Transfer Connections

- Utilizzate per il trasferimento dei dati
- Temporanee
- Basate su HTTP

GNUTELLA 0.4

- Gnutella 0.4: una delle prime proposte di rete p2P completamente decentralizzata

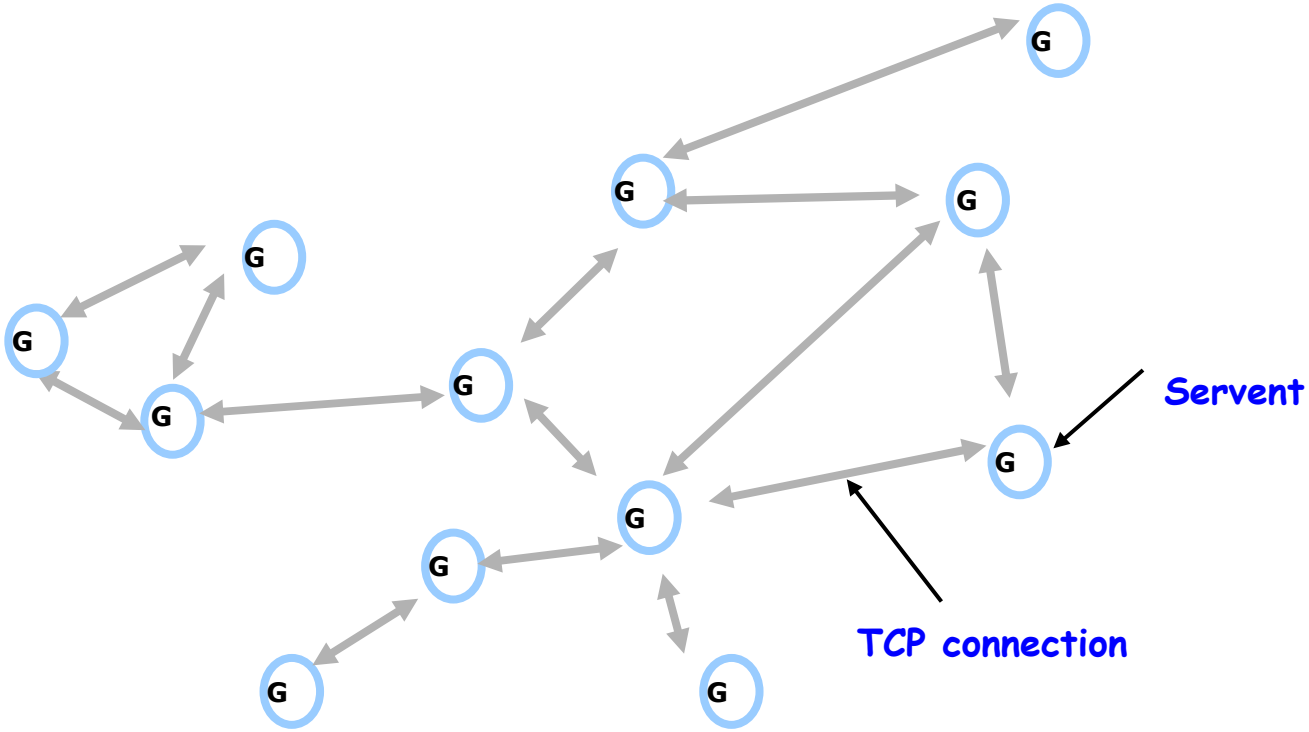
- Documentazione dettagliata del protocollo:

  http://rfc-gnutella.sourceforge.net/developer/index.html

- Un po' di storia:
  - proposto nel 2000 da Frankel e Pepper, sviluppato in non più di un mese
  - Gnutella = GNU(Not Unix, software libero) + Nutella (di cui gli autori andavano matti...)

- Le specifiche del protocollo sono molto ampie: diverse implementazioni disponibili. Le più recenti introducono diverse ottimizzazioni.

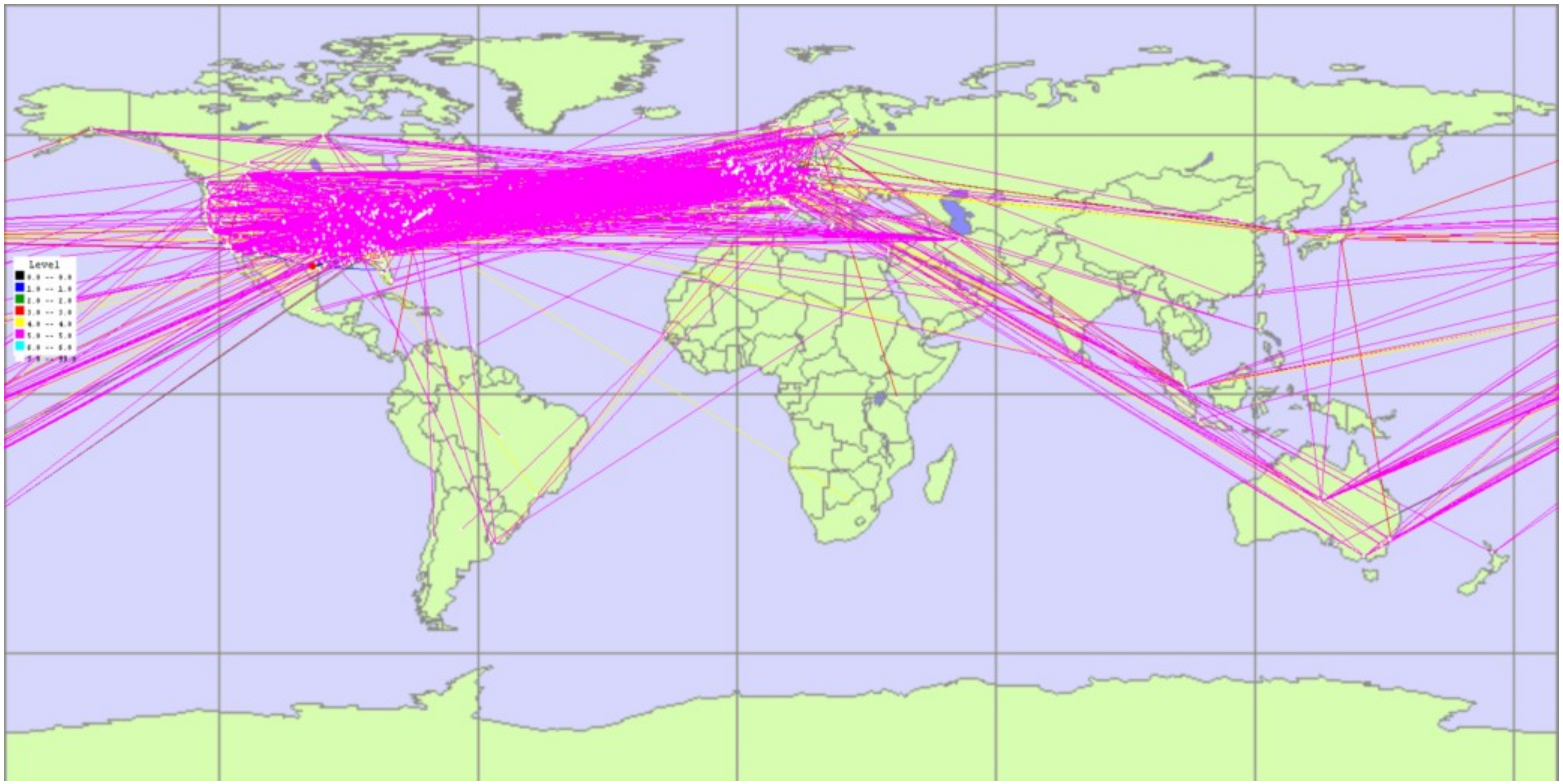
LA RETE GNUTELLA



- I peer sono connessi mediante una rete logica, l'overlay network (rete di copertura)

- Archi= connessioni TCP tra i peers, corrispondono a collegamenti astratti a cui possono corrispondere diversi collegamenti fisici
Es:un arco tra un peer situato negli Stati Uniti ed uno in Polonia

LA RETE GNUTELLA



Topologia della rete Gnutella rilevata
nell'agosto 2002

GNUTELLA: IL PROTOCOLLO IN BREVE

1) Bootstrap:

- Inserimento del peer nella rete mediante individuazione di peers attivi ed apertura di connessione essi.
- Meccanismi di bootstrap basati su
  - beacon (bootstrap) servers: memorizzano una lista di peers spesso attivi sulla rete
  - peer cache: memorizzano liste di hosts contattati in sessioni precendenti
- Evoluzione dei meccanismi di bootstrap in versioni recenti del servlent Gnutella

2) Esplorazione

- La rete viene esplorata mediante l'invio di messaggi di ping/pong (ping message, pong message). Questo consente di scoprire ulteriori peer sulla rete
- Eventuale creazione di connessioni con altri peer

GNUTELLA: IL PROTOCOLLO IN BREVE

3) Sottomissione di queries

- invio delle queries ai peer vicini (query message). I vicini a loro volta inoltrano la query mediante un meccanismo di enhanced flooding, limitato da TTL.

- ricezione di risposte dai vicini selezione della risposta "migliore" (query hit message)

4) Download

- connessione diretta al peer selezionato e download del file ricercato

- scambio dei dei dati mediante protocollo HTTP

GNUTELLA: IL PROTOCOLLO IN BREVE

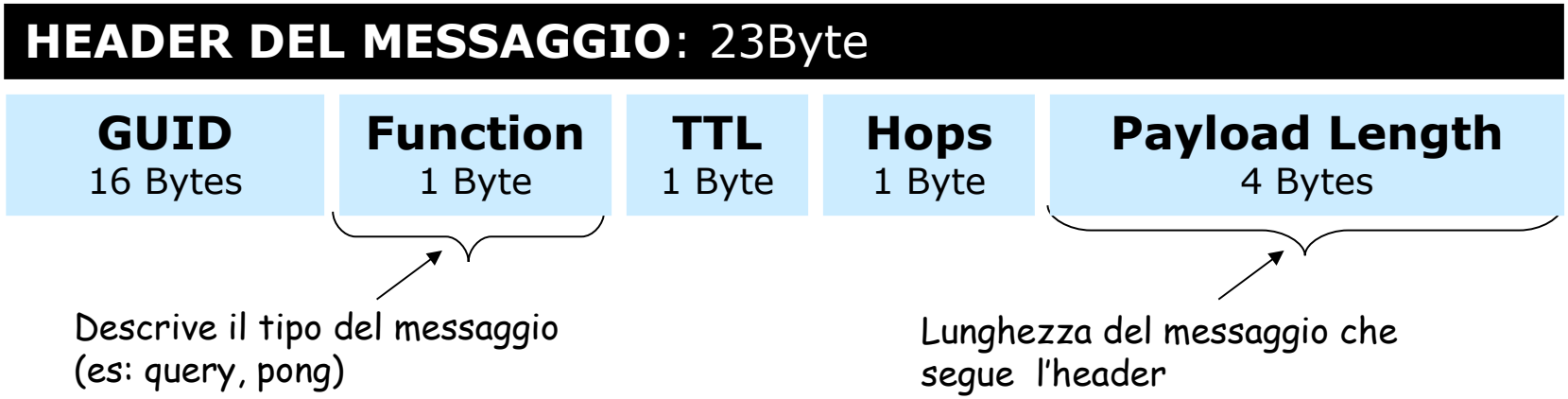
- Principio Base: Enhanced Rooting

- Quando un peer riceve un ping message oppure un query message
  - Decrementa il TTL del messaggio
  - Se TTL=0, il messaggio viene scartato
  - Se il peer ha già inoltrato lo stesso messaggio (utilizzo di identificatori unici di messaggio), il messaggio viene scartato. Questo consente di evitare cicli.
  - Altrimenti il messaggio viene spedito a tutti i vicini, escluso il peer da cui si è ricevuto il messaggio (o ad un sottoinsieme dei vicini)

- Quando un peer riceve un pong oppure un query hit
  - Il messaggio viene inoltrato al peer che aveva spedito il corrispondente messaggio di ping o il corrispondente messaggio di query
  - il peer memorizzale relazioni tra messaggi ricevuti e messaggi inoltrati

GNUTELLA: IL FORMATO DEI MESSAGGI

Struttura Generale dell'Header:



- **GUID: Globally Unique IDentifier** stringa di 128 bits che identifica univocamente il messaggio
- **TTL (Time-To-Live):** quante volte il messaggio può ancora essere inoltrato dai servents prima di essere eliminato dalla rete
- **Hops:** quanti servent hanno già inoltrato il messaggio

$$TTL(0) = TTL(i)+Hops(i) \leq K \text{ (K in genere = 7)}$$

GNUTELLA: IL FORMATO DEI MESSAGGI

PING (0x00)

Non contiene il payload

PONG (0x01)

Port 2 Bytes	**IP Address** 4 Bytes	**Nb. of shared Files** 4 Bytes	**Nb. of Kbytes shared** 4 Bytes

Payload:

- indirizzo IP identifica il peer che invia il pong. Port è la porta su cui esso è disposto ad accettare connessioni
- numero di files che il peer condivide
- numero di kbytes condivisi

GNUTELLA: IL FORMATO DEI MESSAGGI

QUERY (Function:0x80)

Minimum Speed 2 Bytes	**Search Criteria** n Bytes

- **Minimum Speed:**

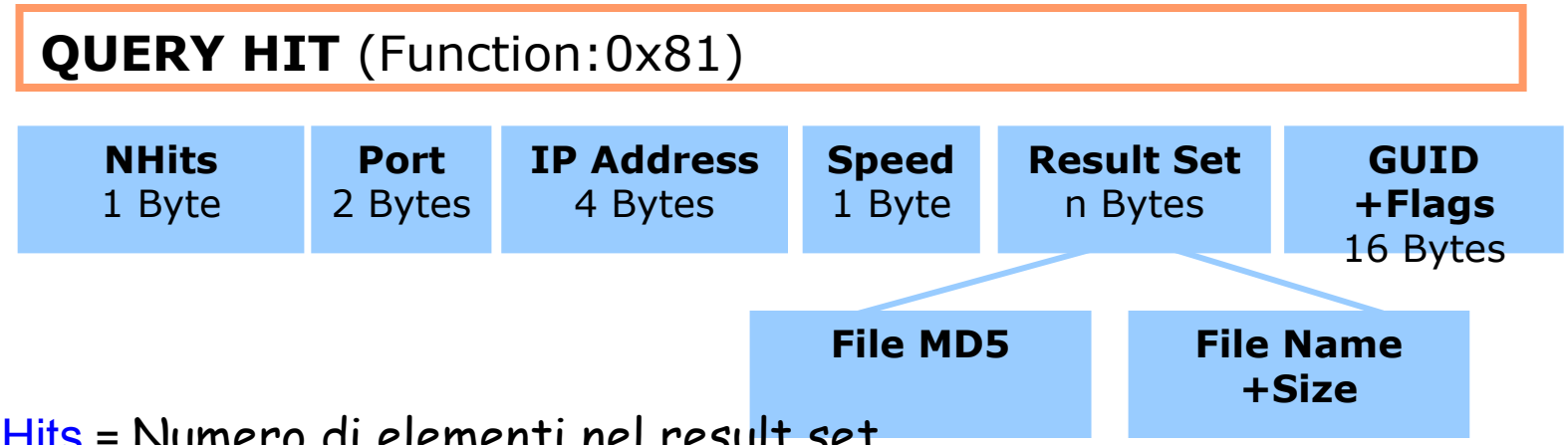
Nelle prime versioni: banda minima (kb/sec) richiesta per i servent che offrono informazioni condivise. Nelle versioni più recenti di Gnutella il campo è stato sostituito da un insieme di flags, tra cui

  - **Firewalled Indicator:** indica che l'host che ha inviato la query non può accettare connessioni perché si trova a **monte di un firewall**. In questo modo un peer che può soddisfare la query evita di inviare un query hit se si trova a sua volta a monte di un firewall.

- **Search Criteria:**

Contiene una stringa costituita mediante la concatenazione di un insieme di keyworld, separate da spazi bianchi.

GNUTELLA: IL FORMATO DEI MESSAGGI



- NHits = Numero di elementi nel result set
- Port + IP Address = Individuano l'host che ha generato l'hit
- Speed = Banda (kb/sec) dell'host che ha generato l' hit
- Result Set = Contiene Nhits elementi e descrive i files che hanno generato l'hit
- GUID+Flags = GUID del nodo che ha inviato il query hit + flags che indicano se tale nodo si trova a monte di un firewall (utilizzati per i messaggi di push)

GNUTELLA: IL FORMATO DEI MESSAGGI

PUSH : (Function 0x40)

Servent Identifier 16 Bytes	**File Index** 4 bytes	**IP Address** 4 Bytes	**Port** 4 Bytes

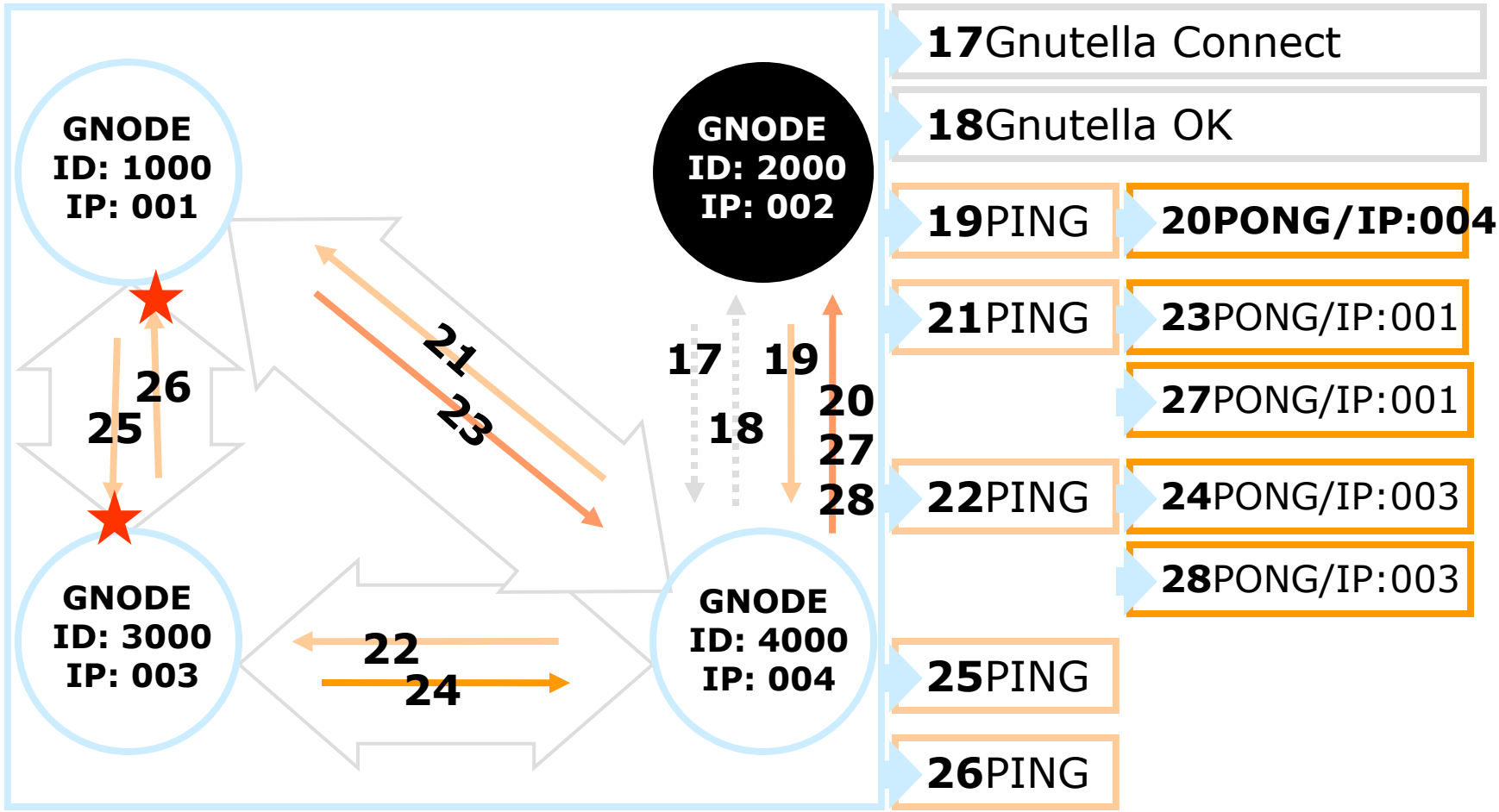
- Servent Identifier: identifica l'host che si trova a monte del firewall (il servent che fornisce il file)
- File Index identificatore del file che si intende scaricare
- IP Address e Port identificano l'host su cui si deve fare il push (il file che ha richiesto il file

GNUTELLA:IL PROTOCOLLO IN BREVE

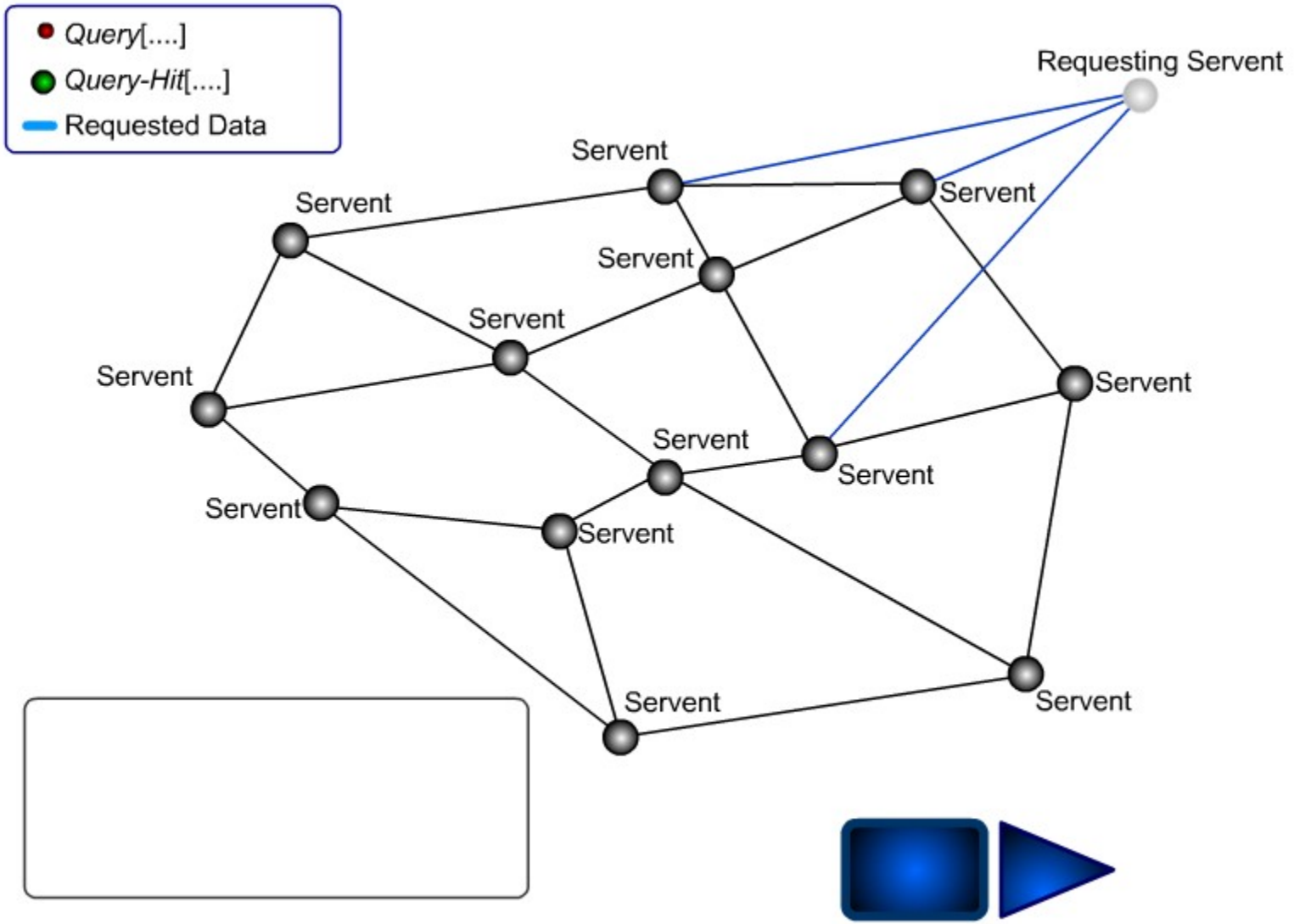
- Il servent Gnutella individua uno o più hosts nella rete (tecniche di bootstrap analizzate in dettaglio in seguito)
- Conosce indirizzo IP e numero di porta di ogni hosts individuati
- Stabilisce una connessione TCP con ciascuno di questi hosts
- Per ogni connessione TCP stabilita
  - Invio di una stringa **GNUTELLA CONNECT/<protocol version> \n\n**
  - Ricezione della risposta **GNUTELLA OK \n\n**
- Il nodo esplora la rete inviando messaggi di ping ai vicini

GNUTELLA: IL BOOTSTRAP

Il nodo 2000 stabilisce una connessione con il 4000



GNUTELLA: IL PROTOCOLLO IN BREVE



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). Il Backward Routing viene individuato mediante gli identificatori unici associati ai messaggi

GNUTELLA: IL PROTOCOLLO IN BREVE

- Perché i messaggi di query hit e di pong vengono inviati mediante backward routing, 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 disattivate, dopo la ricezione di un pong o di un query hit
  - Il backward routing consente di implementare politiche di caching (pong caching).

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:

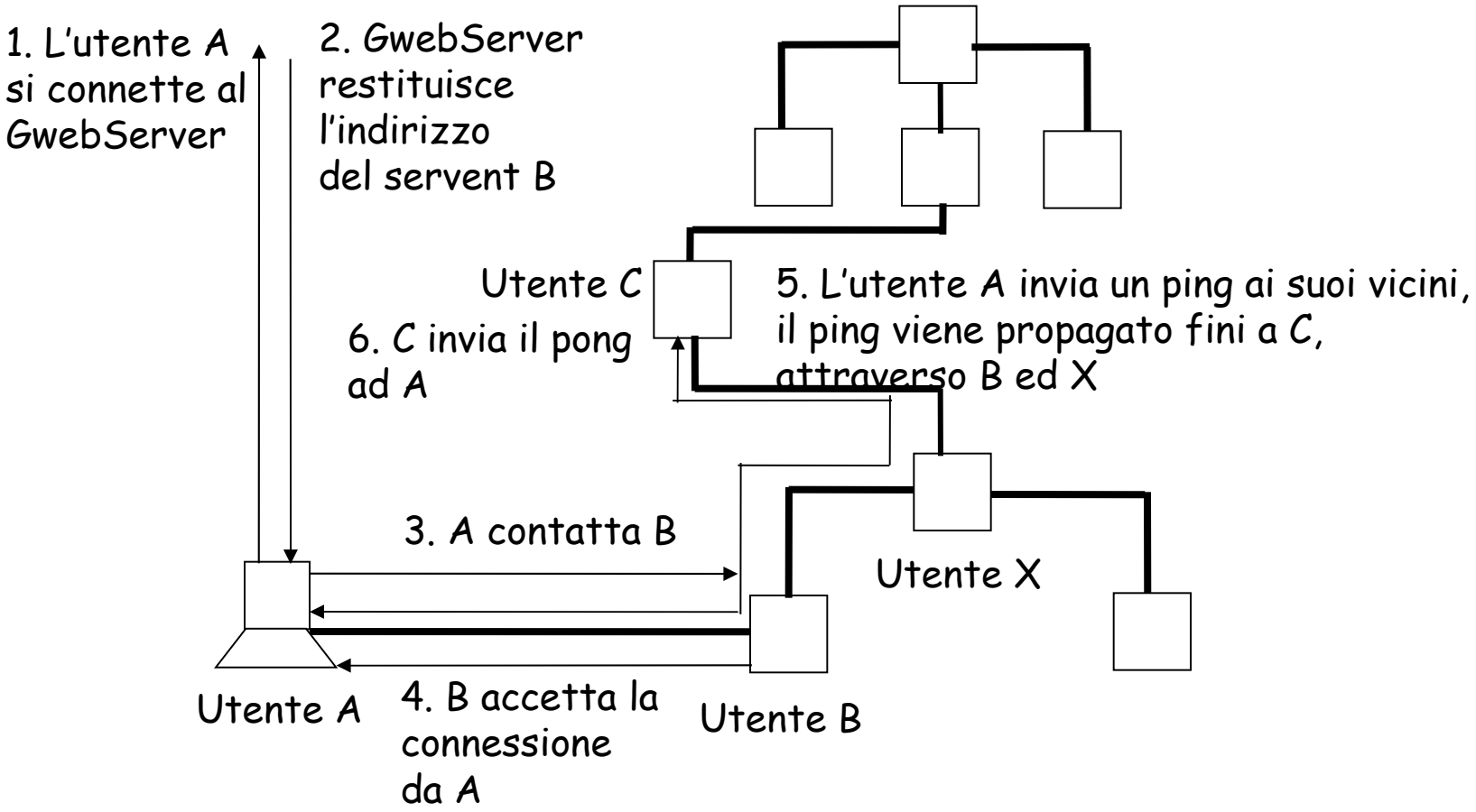
- Out of band methods: gli indirizzi di alcuni servent Gnutella sono pubblicati su alcune pagine web (o si possono ottenere mediante chat). L'utente passa uno di questi indirizzi al servent Gnutella, al momento della attivazione.

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 BOOTSTRAP

- 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

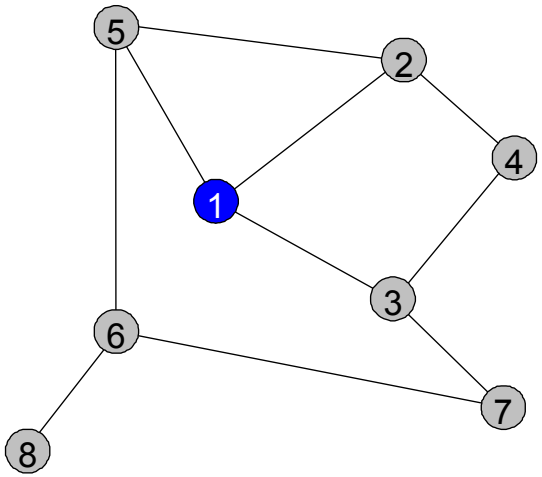
GNUTELLA/0.6 200 OK <cr><lf>
User Agent:Limewire/1.0<cr><lf>
Pong Caching: 0.1 <cr><lf>
X-try:141.2.89.3:6436, 212.43.98.71:6436<cr><lf>

- 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

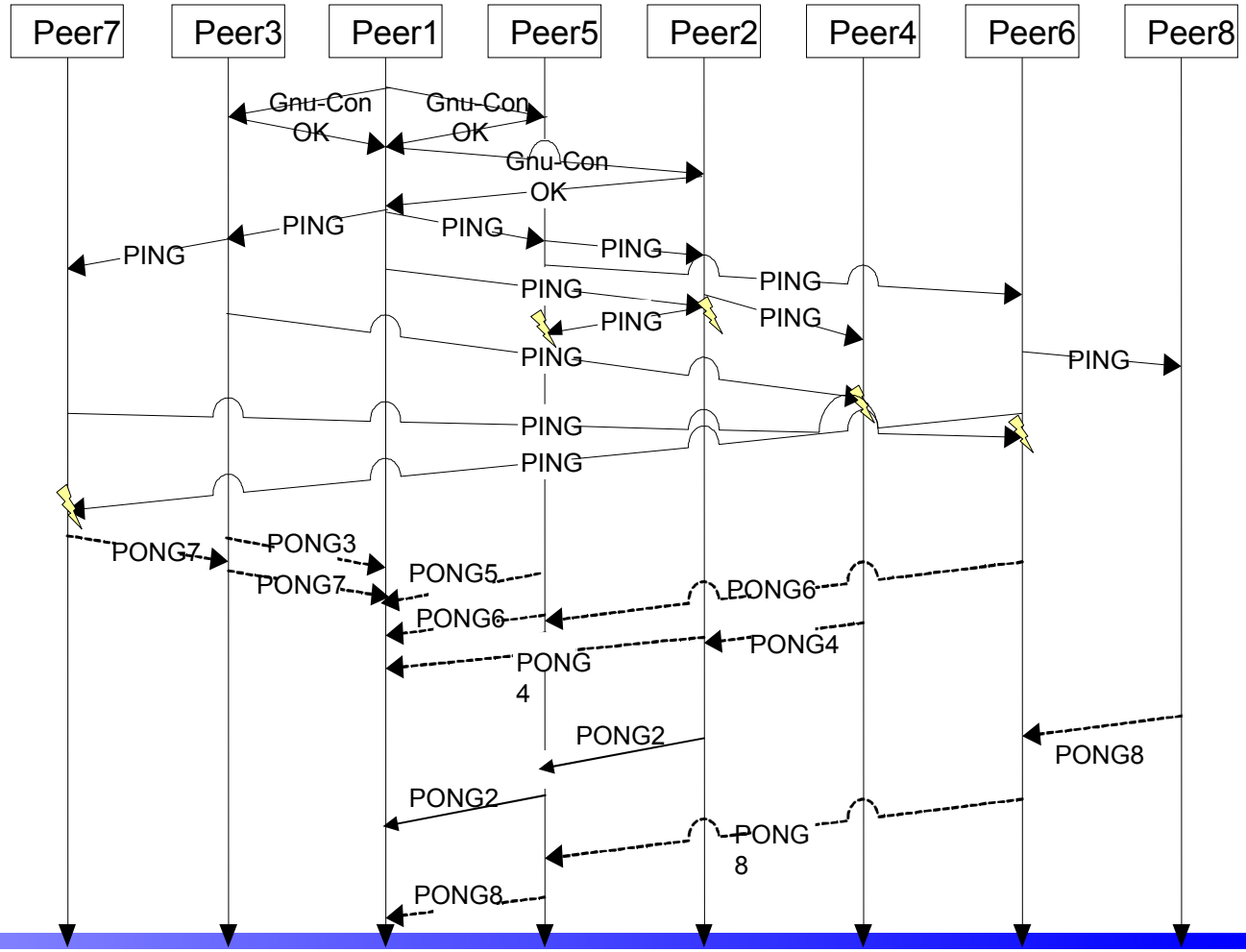
GNUTELLA: ESPLORAZIONE DELLA RETE

Numero di messsaggi per esplorare la rete:

- 6 connessione
- 12 PING
- 12 PONG



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



GNUTELLA: GESTIONE DELLE CONNESSIONI

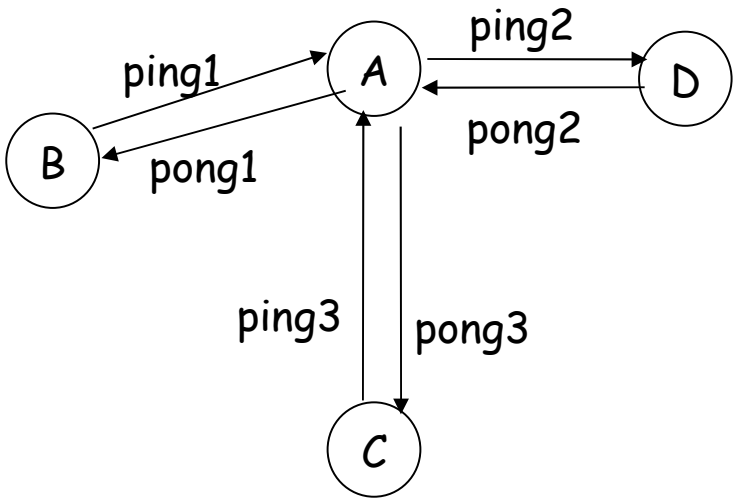
Gestione delle Connessioni

- 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.

GNUTELLA PONG CACHING

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



- A riceve messaggio di ping ping1 da B e genera il messaggio di ping ping2 verso D
- A riceve il pong pong2 da D ed inoltra il pong1 a B. A inoltre memorizza le informazioni contenute nel messaggio pong2 nella propria cache
- Quando A riceve il messaggio ping3 da C utilizza l'informazione memorizzata nella cache per generare il pong3

GNUTELLA PONG CACHING

- Memorizzare per ogni connessione un vettore di messaggi di pong. Per ogni pong memorizzato, si registra Indirizzo IP, Porta, numero di files e di kilobytes condivisi, ed il valore di Hops.

- Quando un servent riceve un messaggio di pong sovrascrive il messaggio di pong memorizzato meno di recente

- 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)

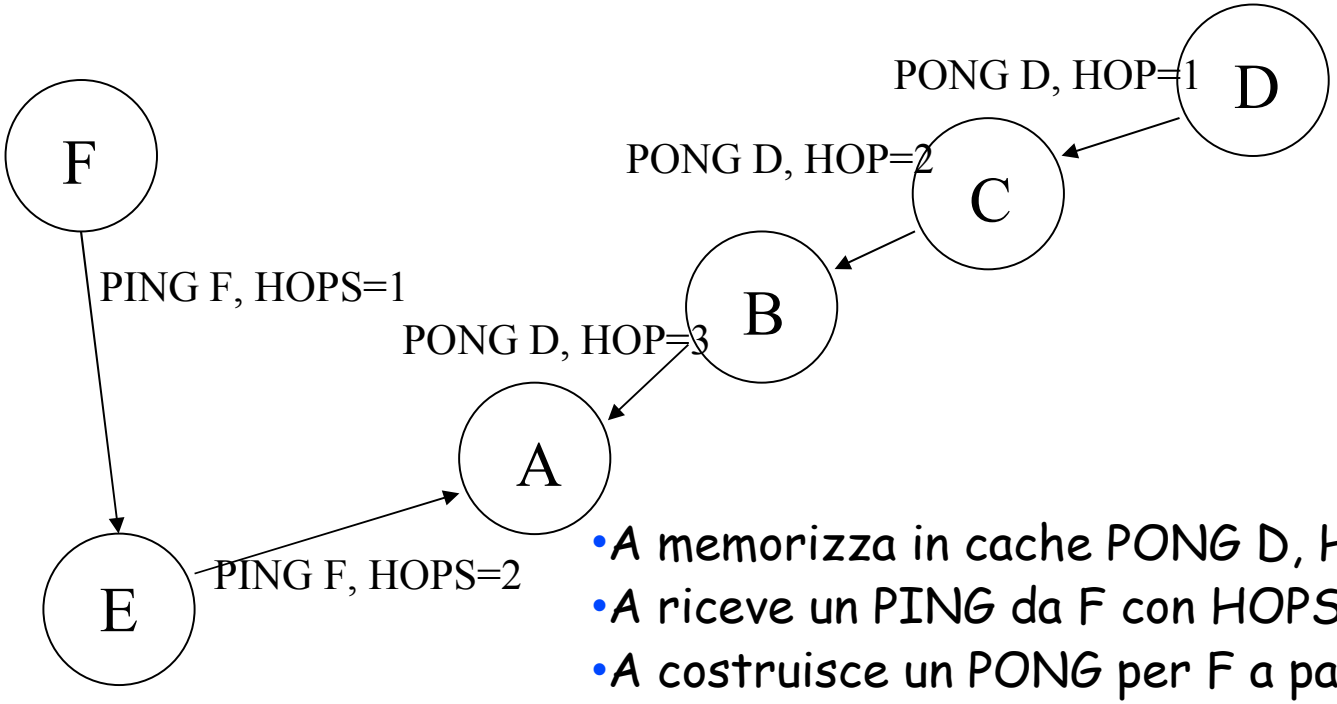
GNUTELLA PONG CACHING

- 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 di risposta al messaggio di ping Pi costruito a partire da un messaggio di pong Pc reperito nella cache:
  - Il GUID 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 PONG CACHING

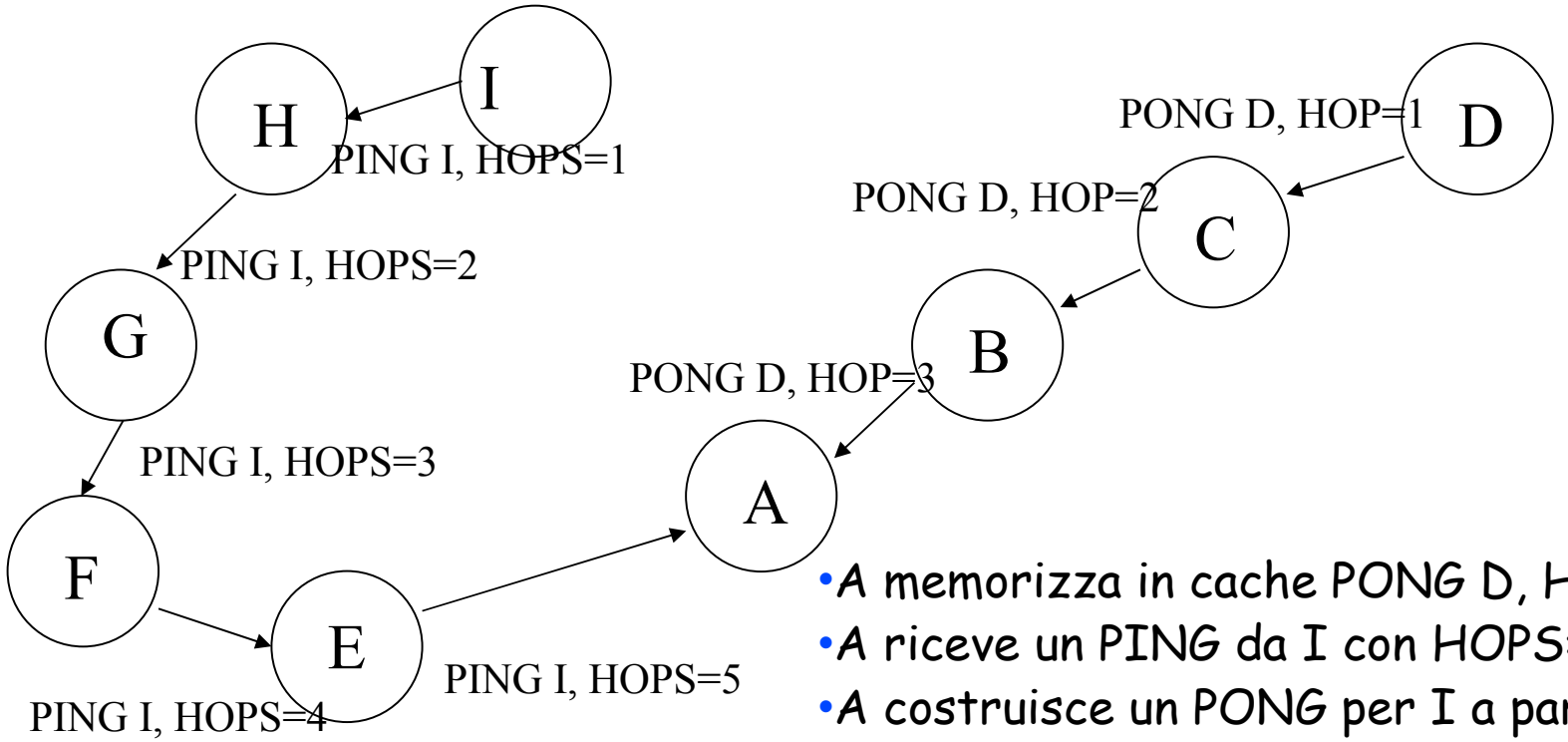


- A memorizza in cache PONG D, HOPS=3
- A riceve un PING da F con HOPS=2
- A costruisce un PONG per F a partire dal PONG D reperito nella propria cache
  TTL(PONG destinato ad F) = 7-3= 4
- Poiche'

  4 >= 2 (HOPS(PING di F))

Il pong viene inviato ad F

GNUTELLA PONG CACHING



- A memorizza in cache PONG D, HOPS=3
- A riceve un PING da I con HOPS=5
- A costruisce un PONG per I a partire dal PONG D reperito nella propria cache
  TTL(PONG destinato ad I) = 7-3= 4
- Poiche'

  4 < 5 (HOPS(PING di F))

Il pong non viene inoltrato

QUERY HIT AND PONG ROUTING

- Per ogni messaggio di query, ping (o push) inoltrato sulla rete, occorre ricordare la connessione da cui il messaggio è pervenuto, in modo da inviare la relativa risposta, query hit o pong, sulla stessa connessione
- Consideriamo i messaggi di query. Il servent S
  - per ogni Query ricevuta memorizza (ad esempio in una hash map), la corrispondenza tra il suo GUID e la connessione da cui la query è stata ricevuto
  - se la Query è stata generata da S imposta a void il riferimento alla connessione.
  - ad ogni Query Hit è associato lo stesso GUID della corrispondente Query
  - quando S riceve il QueryHit, utilizza il GUID corrispondente per effettuare una ricerca nella hash map
    - Se la ricerca restituisce void, la risposta viene inviata all'interfaccia utente
    - Altrimenti il riferimento alla connessione reperito dalla hash map viene utilizzato per propagare il query hit

QUERY HIT E PONG ROUTING

- Un servent non può mantenere la storia di tutti i messaggi ricevuti in una sessione (fino ad un milione di massaggi al giorno)
- Ogni entrata della hash map viene mantenuta per un certo intervallo di tempo (ad esempio 10 minuti), poi viene scartata
- Trattamento diverso per i messaggi di Push

GESTIRE I MESSAGGI DUPLICATI

- Un messaggio può essere recapitato più di una volta allo stesso servent
- Per controllare la duplicazione dei messaggi di query è possibile utilizzare la hash map, in cui si registrano i GUID delle query già inoltrate
- Se il GUID della query è già presente nella hash map, la query non viene propagata
- Soluzione analoga per i messaggi di ping
- Più complessa la gestione della duplicazione dei query hits (solito GUID)
  - Soluzione possibile: calcolare un hash sul result set contenuto nel messaggio
  - Memorizzare in una struttura dati una stringa ottenuta concatenando il GUID del messaggio, il GUId del servent che ha inviato il QueryHIT e l'hash calcolato al passo precedente
  - Confrontare il valore ottenuto con gli analoghi valori memorizzati in precenza

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: MESSAGGI DI PUSH

- Il servent A invia una Query
- B risponde con un messaggio di QueryHit. Supponiamo che B si trovi a monte di un firewall e che non possa accettare connessioni. B inserisce nel query hit il proprio GUID e setta il firewalled flag
- A riceve il QueryHit e ed invia un messaggio di Push a B. Nel messaggio viene inserito l'identificatore di B, un riferimento al file che A vuole scaricare, indirizzo IP e porta di A
- Il messaggio di Push segue lo stesso cammino percorso dal messaggio di QueryHit. Il routing del messaggio di push utilizza l'identificatore di B inserito nel messaggio
- Quando B riceve il messaggio di Push, apre una connessione verso A usando l'indirizzo IP e la porta contenuta nel messaggio di Push ed invia il file richiesto da A

GNUTELLA: IL CONTROLLO DEL FLUSSO

- Controllo del flusso a livello applicazione: garantisce di non sovraccaricare le connessioni aperte, uso delle code associate alle connessioni
- Controllare la dimensione delle code
- Stabilire una politica di scelta dei pacchetti da inviare nel caso di connessioni sovraccariche
- Se la dimensione della coda supera una soglia prefissata, si passa in una modalità di tipo flow-control
- Diverse implemetazioni proposte per il controllo del flusso

GNUTELLA: IL CONTROLLO DEL FLUSSO

Politica per il controllo del flusso

- Scartare tutte le queries provenienti da una connessione congestionata. L'accettazione di una query implica l'invio di un insieme di risultati sulla stessa connessione (la cardinalità di questo insieme può essere potenzialmente elevata)
- 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

GNUTELLA: IL CONTROLLO DEL FLUSSO

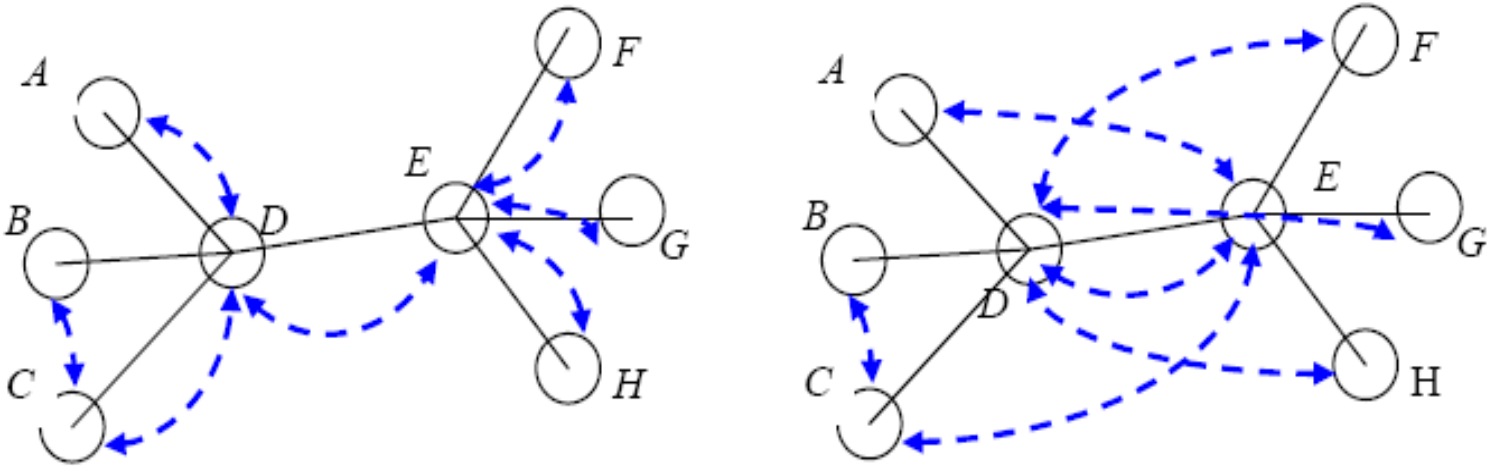
Algoritmi più sofisticati per il controllo del flusso

- Favorire le risposte a query che hanno trovato meno hits (files meno popolari) rispetto a quelle che hanno prodotto molti hits (files molto popolari)
- Esempio: 10 repliche per la query Q1, 1000 repliche per la query Q2. E' meglio inoltrare un query hit per Q1
- Implementazione: ordinare le code relative ai query hits in base al volume di repliche utilizzando il GUID del messaggio
- Utilizzare code diverse per i diversi tipi di messaggi

GNUTELLA 0.4: PROBLEMI

- Problemi: La overlay network può differire notevolmente dalla struttura della rete fisica
- Zig Zag Routing: Esempio servent collocati in Europa ed negli States possono rimbalzare il messaggio più volte attraverso l'Atlantico
- Utilizzo di una gran quantità di banda tra Europo e States
- Ritardi di comunicazione
- Proposta: adattare la topologia dell'overlay network a quella della rete fisica

GNUTELLA 0.4: PROBLEMI



- i collegamenti tratteggiati in blu definiscono l'overlay network
- i collegamenti 'continui' in nero defiscono la struttura della rete fisica
- nell'overlay di sinistra, un messaggio inviato da A raggiunge tutti gli altri nodi dell'overlay attraversando il linkD-E una sola volta
- nell'overlay di destra, un messaggio inviato da A deve attraversare il link fisico d-E per 6 volte per raggiungere tutti gli altri nodi della rete

GNUTELLA 0.4: PROBLEMI

Problema principale: bassa scalabilità

- Il numero di pings e di queries cresce in modo esponenziale ad ogni hop
- Esempio: un nodo invia un messaggio di query con TTL=7 (valore di default per la maggior parte delle implementazioni), ogni nodo lo propaga a x vicini (esempio x=4, n=7)
  - il numero totale di queries propagate sulla rete è circa (vedi serie geometrica):

    $\Sigma_{n=0,7}\ x^n=(4^{n+1}-1)/(4-1)=(4^{7+1}-1)/(4-1)=21845$

  - Il numero di risposte dipende dalla popolarità del file richiesto
- Soluzione
  - Introduzione di appocci ibridi basati su superpeers
  - Definizione di meccanismi di caching