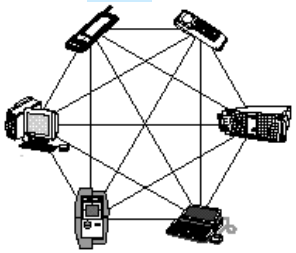


Parte 5: Sistemi P2P

Corso di: Sistemi Distribuiti
Lauree in: Ingegneria Informatica,
delle Telecomunicazioni ed Informatica di Scienze

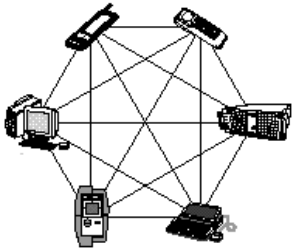
Prof. Paolo Nesi

Department of Systems and Informatics, University of Florence
Via S. Marta 3, 50139, Firenze, Italy
tel: +39-055-2758515, fax: +39-055-2758570

DISIT Lab, Sistemi Distribuiti e Tecnologie Internet

http://www.disit.dinfo.unifi.it/

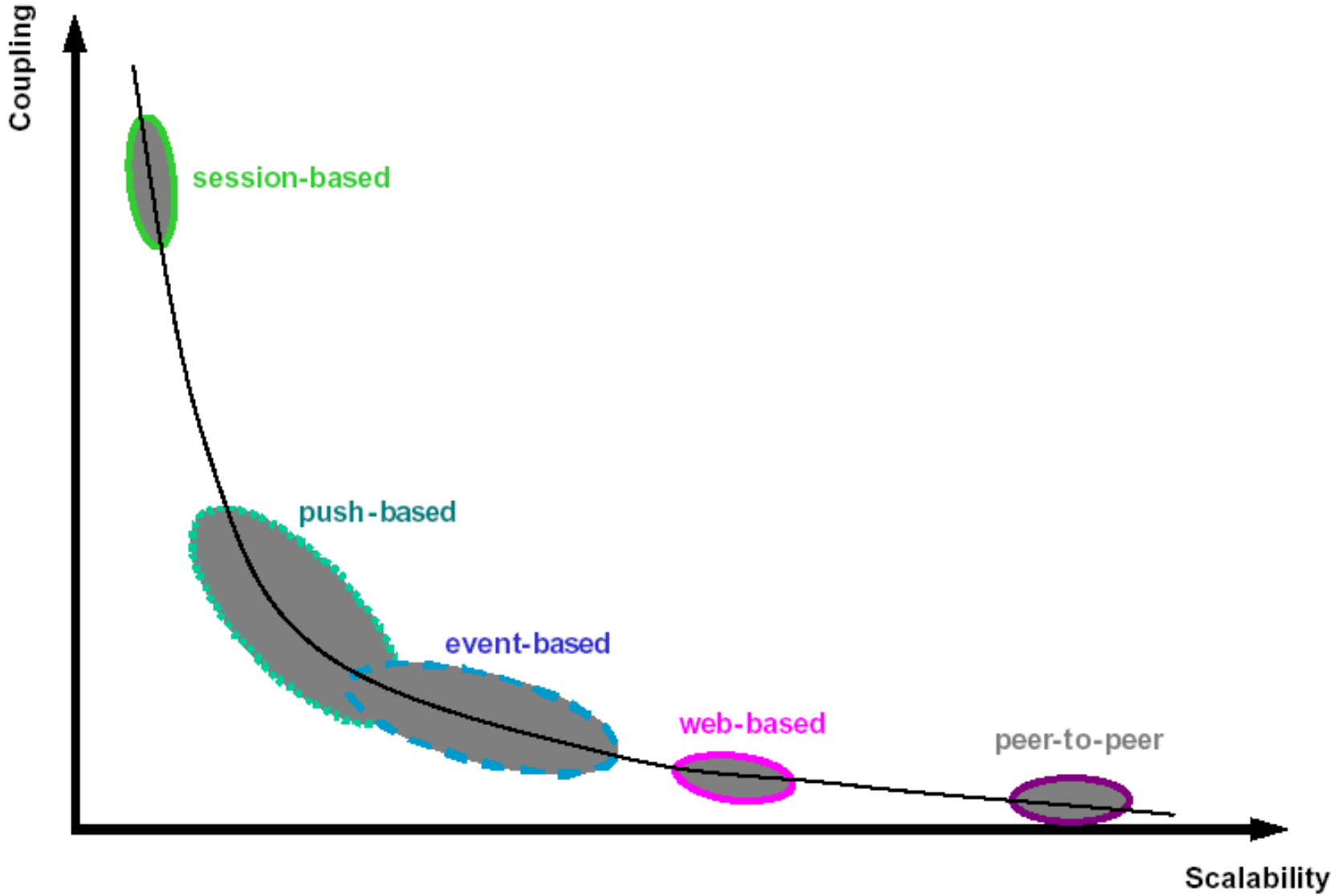
paolo.nesi@unifi.it
http://www.disit.dinfo.unifi.it/nesi

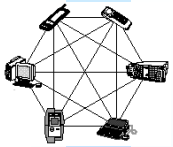


Sistemi P2P

- Aspetti Generali, Applicazioni
- Requirements
- Architecture P2P e caratteristiche
- Ricerche e download multisorgente, BTorrent
- Reti P2P in Overlay
- Esempi: Skype, P2P
- Esempio: P2P distributed trust

Coupling and Scalability

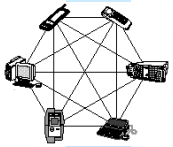




Application areas

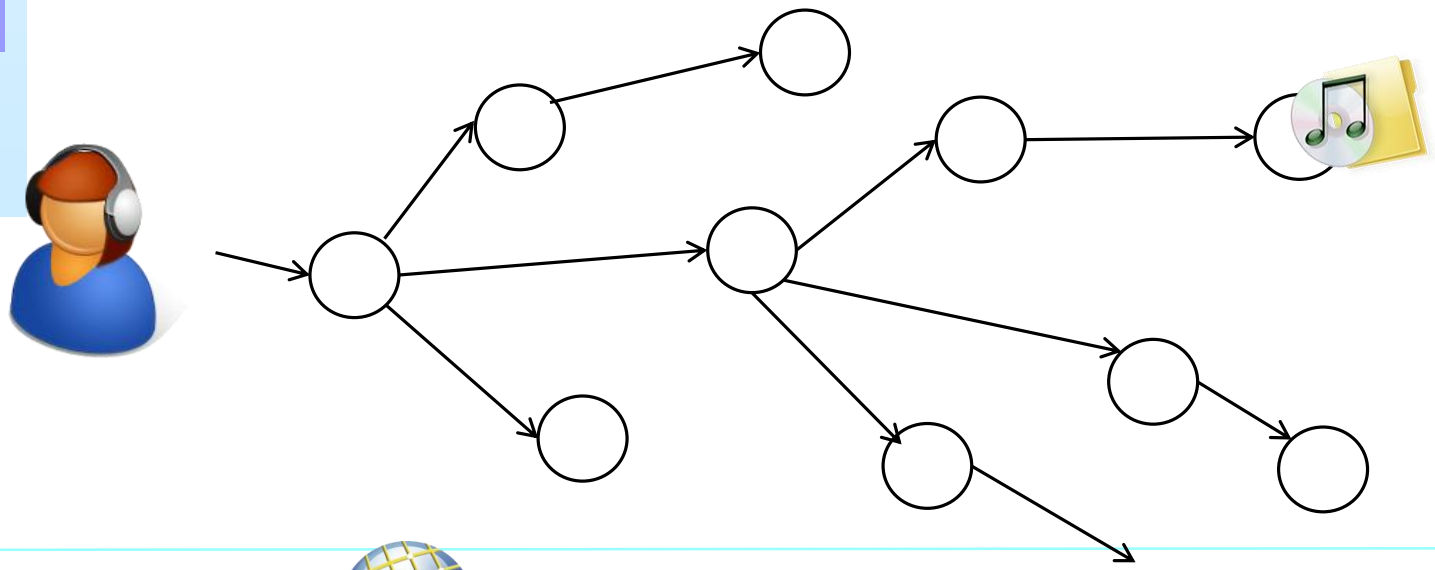
- **Content and Resource sharing**
  - Network-wide file/document sharing (napster, eDonkey, Gnutella, Freenet, piratebay, emule, etc.)
  - VOIP: Voice Over IP
  - P2P CDN: Content delivering network
  - P2P VOD (Video on demand), P2PTV, WebTV also in STB

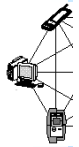
- **Distributed computation more GRID than P2P**
  - Internet-based (e.g. United devices, entropia)
  - Intranet-based (www.datasynapse.com, NetBatch of Intel)
  - Web testing (e.g., United devices)
  - Esempio: gridella, etc.…
  - Resource sharing: seti@home, aids@home, folding@home



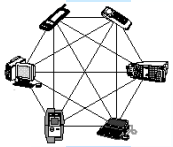
P2P Applications for file sharing

- Napster, Gnutella, Freenet, Kazaa
- Emule, Emule Plus: both based on kademlia
- Mojo Nation
- BitTorrent (Azureus client)
  - BT based: PirateBay, Suprnova, isoHunt, TorrentSpy
- **Shareaza** supports protocols like: Gnutella, Gnutella2, eDonkey Network, BitTorrent, FTP and HTTP



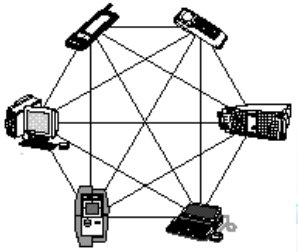


Traffic Category	Description	Examples
Storage	Large data transfers using the File Transfer Protocol or its derivatives. Services that provide file-hosting, network back-up, and one-click downloads	FTP, Rapidshare, Mozy, zShare, Carbonite, Dropbox
Gaming	Console and PC gaming, console download traffic, game updates	Nintendo Wii, Xbox Live, Playstation 2, Playstation 3, PC games
Marketplaces	Marketplaces where subscribers can purchase and download media including applications, music, movies, books, and software updates	Google Android Marketplace, Apple iTunes, Windows Update
Administration	Applications and services used to administer the network	DNS, ICMP, NTP, SNMP
Filesharing	Filesharing applications that use a peer-to-peer or Newsgroups as a distribution models	BitTorrent, eDonkey, Gnutella, Ares, Newsgroups
Communications	Applications, services and protocols that allow email, chat, voice, and video communications; information sharing (photos, status, etc) between users	Skype, WhatsApp, iMessage, FaceTime
Real-Time Entertainment	Applications and protocols that allow "on-demand" entertainment that is consumed (viewed or heard) as it arrives	Streamed or buffered audio and video (RTSP, RTP, RTMP, Flash, MPEG - OTHER), peercasting (PPStream, Octoshape), specific streaming sites and services (Netflix, Hulu, YouTube, Spotify,)
Social Networking	Websites and services focused on enabling interaction (chat, communication) and information sharing (photos, status, etc) between users	Facebook, Twitter, Linkedin, Instagram
Tunneling	Protocols and services that allow remote access to network resources or mask application identity.	Remote Desktop, VNC, PC Anywhere, SSL - OTHER, SSH,
Web Browsing	Web protocols and specific websites	HTTP, WAP browsing



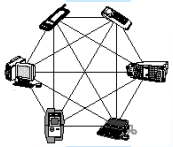
Definition of Peer

- The peer is a node client of the P2P network
  - Each client peer has many files
  - Some in download and/or uploads
- The ***peer is a single thread***, process of download and/or upload, such as in BitTorrent Terminology
  - Each client node has many peers, typically no more than 5/10 at the same time.
- We can have a network that at a given time instant may have
  - 4Mpeers, 1.2Mfiles and 890Kusers
  - Some are seeders the other are passively reading only !



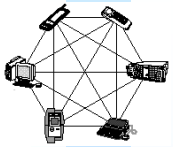
Sistemi P2P

- Aspetti Generali, Applicazioni
- Requirements
- Architecture P2P e caratteristiche
- Ricerche e download multisorgente, BTorrent
- Reti P2P in Overlay
- Esempi: Skype, P2P
- Esempio: P2P distributed trust



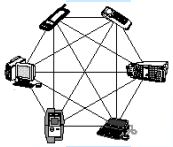
P2P Main requirements

- **Creation of the P2P community of Peers/clients**
  - Discovering of the Peers/clients to reach the resources
- **THUS: Discovering resources/services**
  - Resources may be objects, files or disk space, or computational power, users, etc.
  - Allocation of resources/objects into the P2P network
    - Global Unique ID, GUID for the objects
  - Indexing of the resources into the P2P network
    - Customization of query for getting information
- **Managing updates in the information shared**
  - In the information requested only
  - Removing obsolete files and/or references
    - it is not always possible in P2P solutions in which it is tracked who has downloaded the file, or has the reference
  - Notification of changes in the downloaded files, in the accessed resources, etc.
    - Versioning, replacement
    - Notification to all peers that have downloaded, please stop providing the last version.
    - Again: also in this case the system has to keep trace of who has the file or the reference



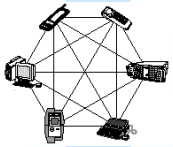
P2P Main requirements

- **Scalability of the P2P solution**
- From 1 peer/resource/user to Millions and Millions
  - How is the capability in penetrating the network, Intranet to internet,
    - Discovery, query, versioning, maintenance, etc.
    - In intranet UTP can be used
    - In Internet UTP CANNOT be used
    - Peers need to perform the boot of the P2P network in Internet
  - Intrinsic limits of models,
    - for example a limit on the code for unique ID for files, users, etc..
    - How it may graw ?
  - how is the costs of the model to grow in terms numbers of Peers??
    - How many servers are needed ?
    - Which networks capability, bandwidth, they need ?
    - Which is the velocity that the graw may sustain ??



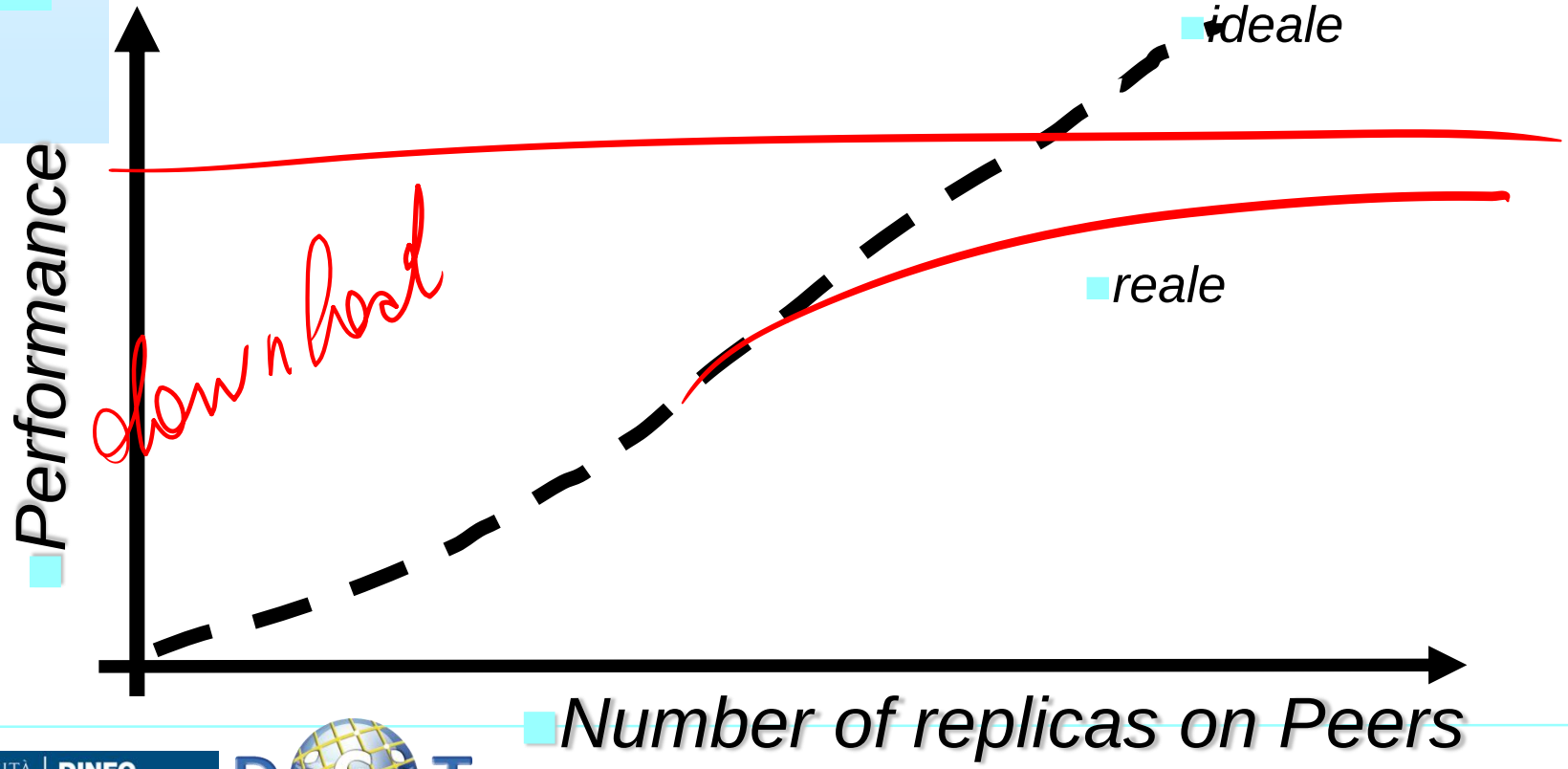
P2P Main requirements

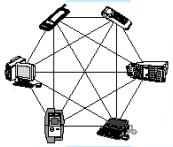
- **Performance**
  - Analysis and control of the network connections
    - Banda per esempio in termini di Mbps/Kbps
    - massimo numero di connessioni apribili/attive in ingresso (download) ed uscita (upload)
    - From the peer to a set of reference peers/servers
  - Measuring CPU features:
    - fixing % of free CPU reserved
  - Space on the HD disk:
    - space reserved, (maximum) space accessible, effectively free space used in the shared files, etc..
  - Max Number of shared files, opened connections:
    - reserved and maintained visible
  - Time to download, time to start the download
    - Time to perform the download, start-end time
  - etc.



Performance of P2P solutions

- Grafo dell'andamento delle prestazioni (download rate) in funzione del numero dei peer che hanno un certo file, misurato in un certo punto della rete.
- L'obiettivo della distribuzione e' arrivare a superare una certa soglia nel minor tempo possibile. Il superamento della soglia di seeding mi garantisce delle prestazioni ragionevoli in termini di capacità di prestazioni per/gli l'utente/i e la diffusione del file nella rete.
- La saturazione e' dovuta ai limiti di banda del client

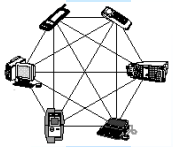




P2P Main requirements

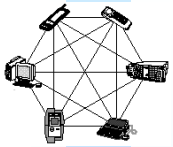
- **Security and Trust of users**
  - Authentication of users
    - Identification of the user in terms of name/surname, etc., or in terms of a simple UID, watermark
      - Knowledge of who has posted the file
  - Privacy is typically preferred by P2P users
    - Privacy vs Authentication of users and Peers

- **Security and Trust of Content**
  - Data/file/object certification: consistency
    - Authentication lead to higher level of trustiness
    - Trust of metadata and data, certification of content:
    - per verificare/riconoscere la firma del content,
    - garantire la consistenza fra metadati e dati
  - Authorization to delivering and use content
    - Controllo dei diritti, Digital rights management
    - gestione dei diritti/rights, licenze, etc.



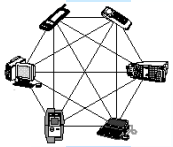
P2P Main requirements

- **Robustness, Fault tolerance**
  - Robustness with respect to eventual fault of
    - Single peers
      - Interruption of downloads
      - Interruption of query/interrogation service
      - Interruption of instradation service (see Routing Overlay)
    - SuperNodes that permit the indexing and/or the boot of the P2P community
    - Network problems, turn off/on of the peers
- **Definition of solutions for recovery from failure**
  - Recovering the interrupted download of file when it is
    - monolithic: total restart
    - segmented: restart of the segment
    - Choosing a different Peer from which the download can be performed
  - Duplication of resources (usage of strategies for duplication)



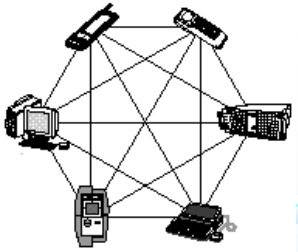
P2P Technological Challenges

- **Architecture**
  - Usable on different platforms
    - ➔ Java has been the most selected
  - Interoperability between the applications built on different P2P infrastructures (diff. protocols, languages, etc.)
  - Controllability of Peers
    - ➔ Monitoring of user/peer behavior
  - Performance, Scalability
  - Fault tolerance
  - Security: Privacy, Trust of content and of Peers



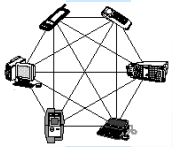
P2P Technological Challenges

- **Architecture**
  - Efficient in joining the network, discovery if needed
  - Efficient setup of the P2P community
  - Query/search of resources
    - Complex queries: such as those based on SQL or RDF, based on semantics: title xx, author YY, ….
    - Connection with local databases: ODBC, JDBC, etc.
  - Deleting of files
    - Removing from the network
  - Changing of files
    - Notification of changes in the files posted/changed on the network
    - versioning
  - QOS: Quality of Service
    - performance in *querying* and in the *download* for B2B and for B2C



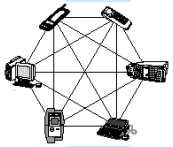
Sistemi P2P

- Aspetti Generali, Applicazioni
- Requirements
- Architecture P2P e caratteristiche
- Ricerche e download multisorgente, BTorrent
- Reti P2P in Overlay
- Esempi: Skype, P2P
- Esempio: P2P distributed trust



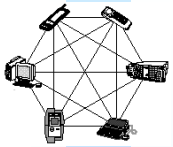
Architetture P2P

- Concentrate, centralized
- Distribuite, Distributed, decentralized
- Hierarchical or hybrid



Centralized P2P Architectures

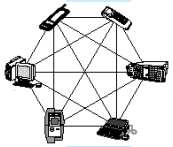
- **Concentrated, centralized**
  - One server and N peers; in some cases, more servers
  - Example: Napster (central index)
- **Also called "Server-based" which may support:**
  - Login, registration of peers to the central server
  - Boot: performed asking to the server to get list of nodes
  - Search: performed asking to the server
  - Collection of data, index, query, etc.
    - Table to know where the files (their replicas) and their segments are: obj45: n3, n4, n56, n78
- **Server Problems: fault, size, performance, cost...**
- **Gli scambi dei file/risorse possono essere:**
  - Centralised or P2P, multisource



Napster Pros and Problems

- **Veloce per l'entrata nella comunità** tramite server centrale o server locali ma sempre tramite server
- Anonimità degli utenti

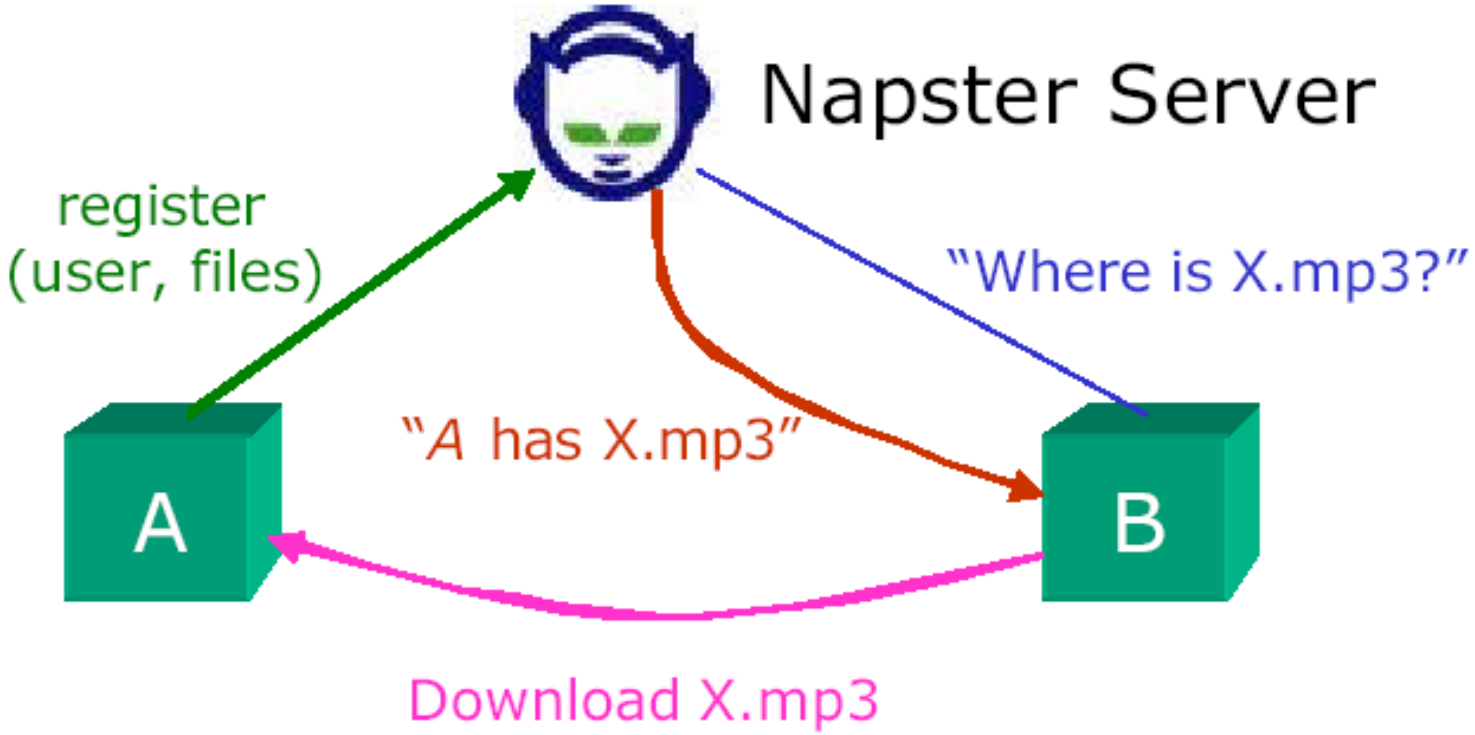
- **Nessun controllo** su dati coperti da IPR (intellectually property right) e questi venivano centralizzati (come indice) in modo non autorizzato
  - Questo problema e' stato risolto nella versione attuale non molto diffusa ed apprezzata, dalla massa
- **Sicurezza:**
  - Contenuti non certificati
  - Utenti non autenticati
  - Non accesso a database, query molto semplici
- **Protocollo proprietario**, filtrato da firewal
- **Scarsa scalabilità**
  - Costi elevati di gestione



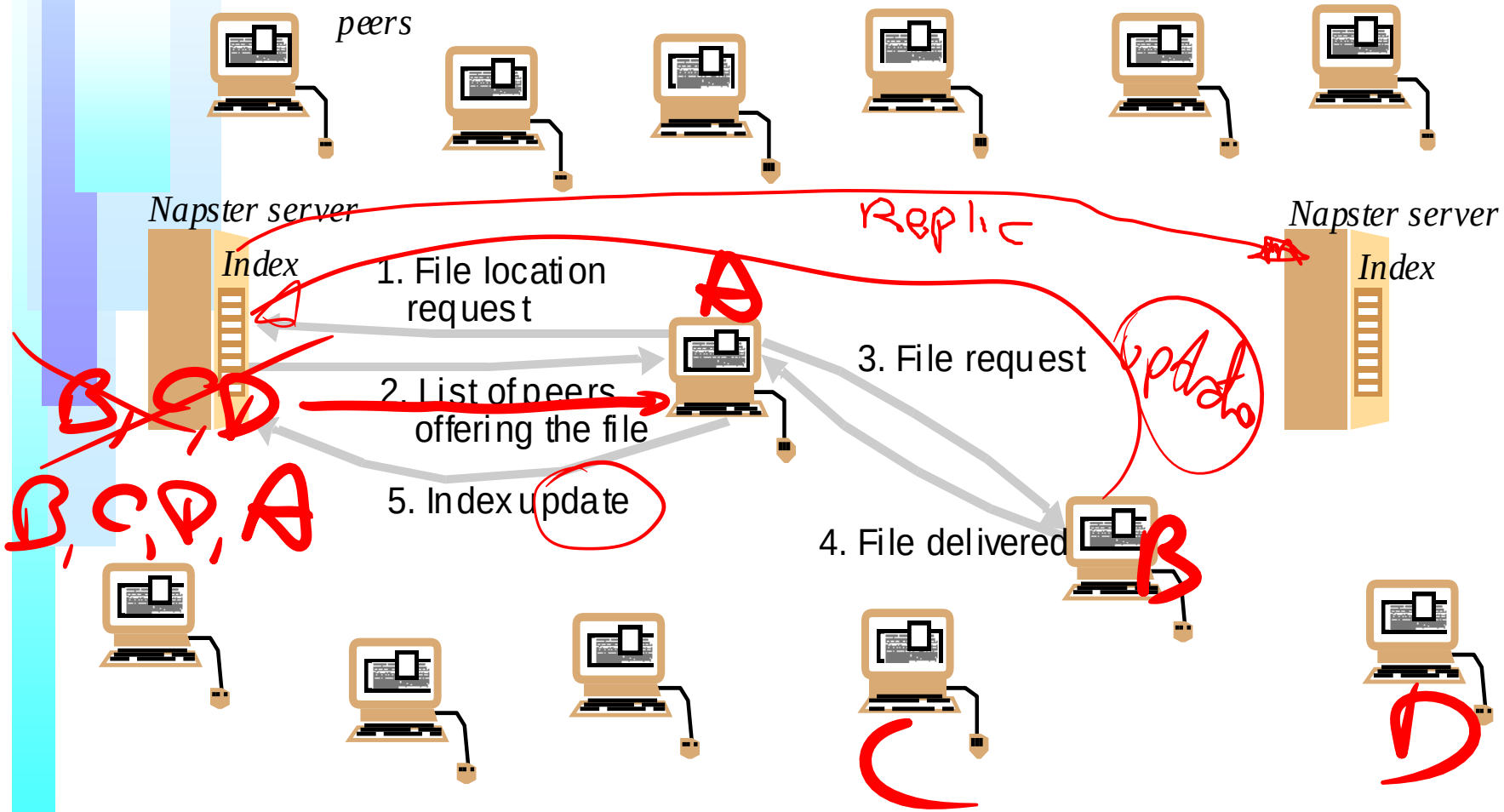
Napster

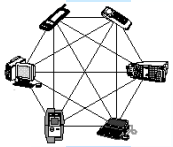
- Architettura centralizzata per la condivisione di file
- Query centralizzata
- Copie dei contenuti centralizzate, in parte
- Al crescere del numero degli utenti e' necessario aumentare le prestazioni e lo spazio disco del server centrale che dai il servizio
- Una o piu' soluzioni:
  - Fare un cluster di server
  - Duplicare le risorse su piu' server
- Aumento dei costi

Napster: search files



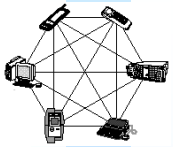
Napster: peer-to-peer file sharing with a centralized, replicated index





Distributed P2P Architecture

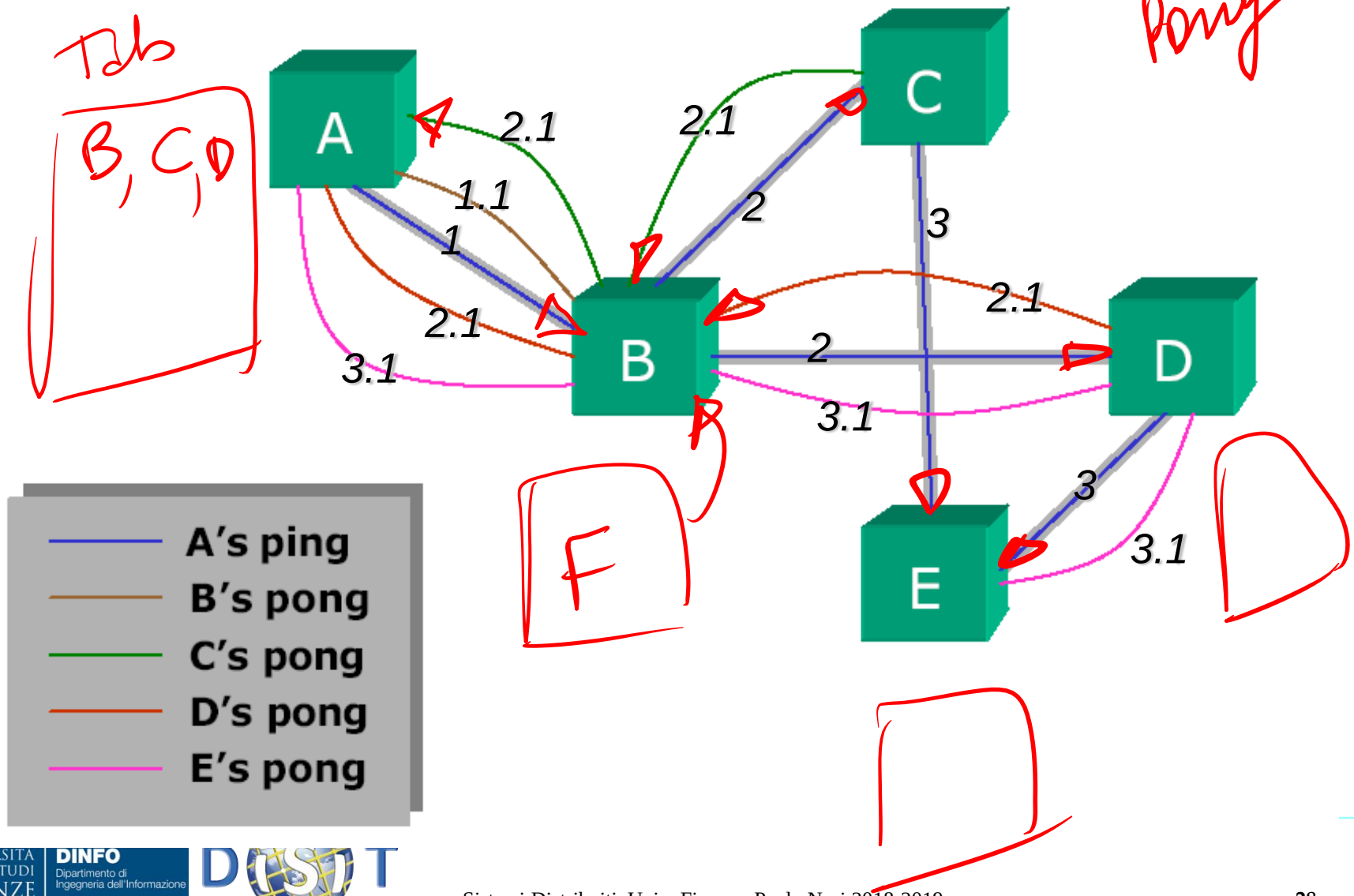
- **Distributed, decentralized**
- **Also called Pure P2P networks**
  - N peers, all identical
  - Example: Gnutella (gnutella hosts), freenet
  - Boot: massive discovery, highly complex/net-costs
  - Search: fully distributed!, high complexity
  - No problems of fault
    - redundancy of information and services
  - The most common problems:
    - Low performance on search and discovery (distributed), etc.
    - No administration, no certification
    - No control on the network



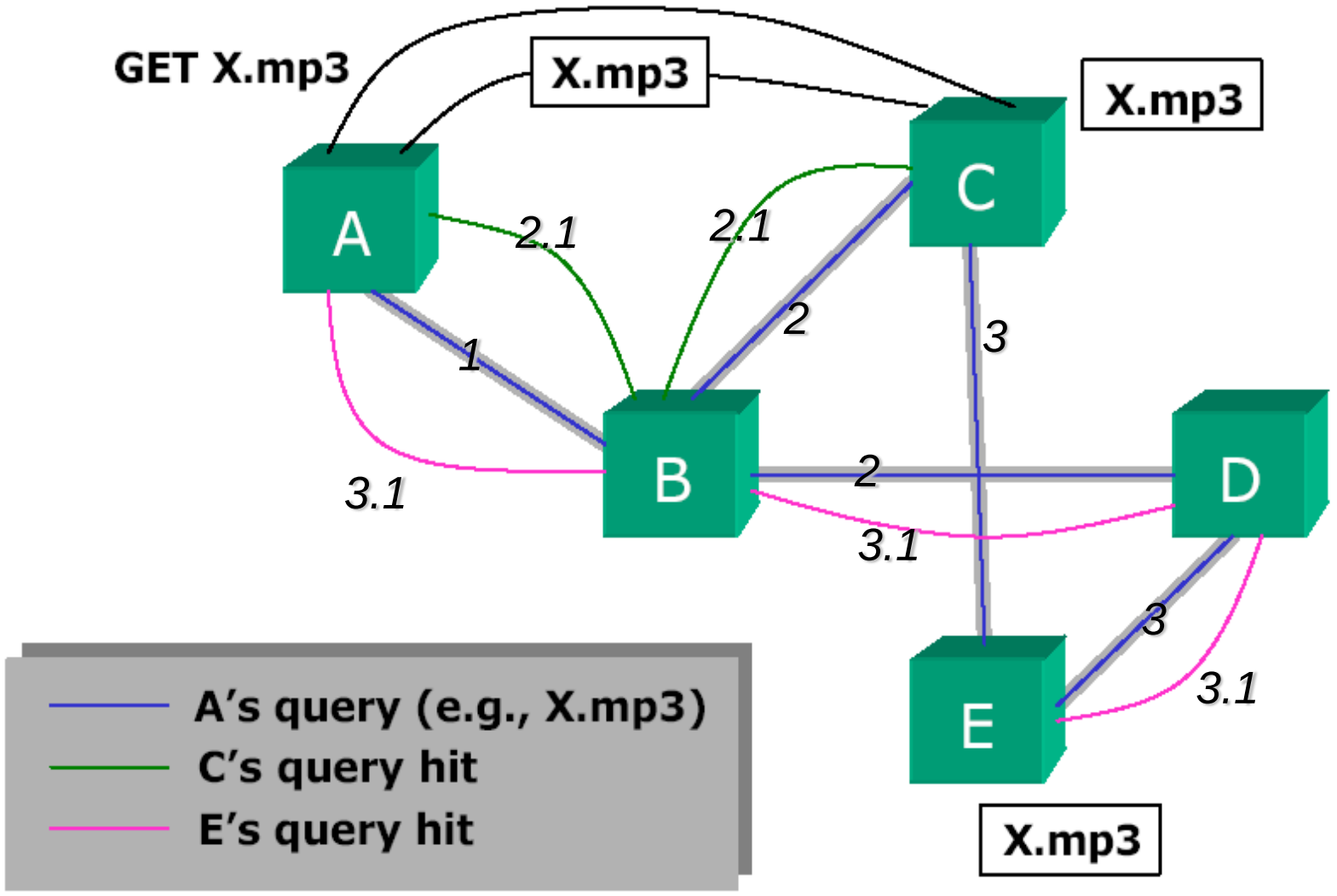
Gnutella

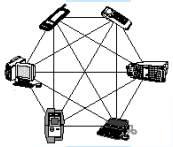
- March 2000, Molto semplice
- Meccanismi di distribuzione e monitoraggio dei file in HTTP
- Non vi sono meccanismi di sicurezza
  - Non e' possibile autenticare gli utenti
  - Gli oggetti non sono certificati
  - *metadati e contenuti possono non essere consistenti*
- http://www.gnutelliums.com
- www.gnutella.com
- Several implementations of Gnutella clients
  - For example: limewire,

Gnutella: discovering peers



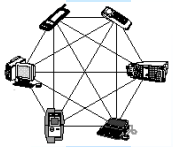
Gnutella: searching (via routing)



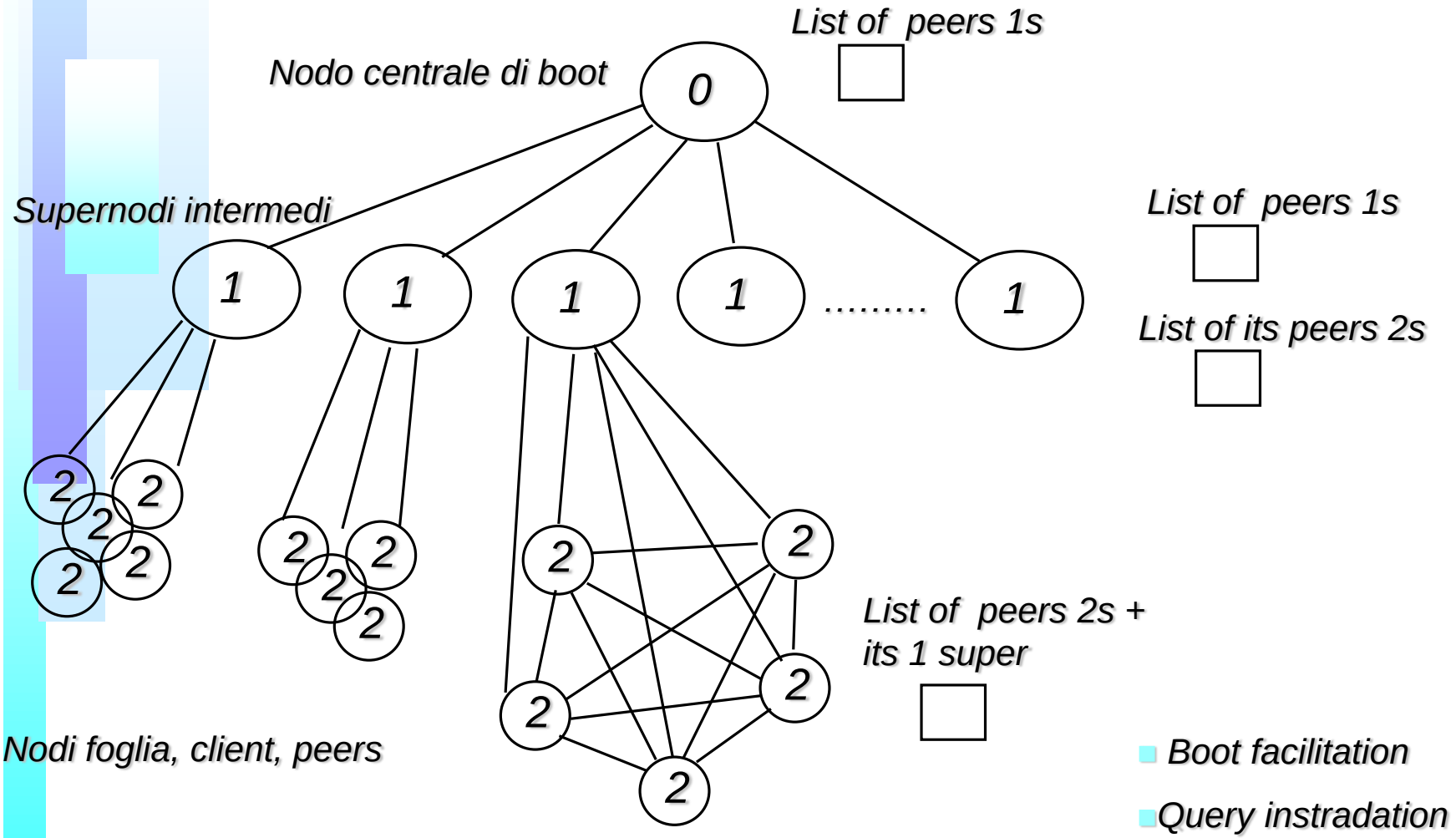


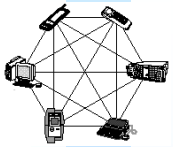
Hybrid P2P Architetture

- **Hierarchical, hybrid**
  - Mix of centralized and decentralized
  - N peers *not all identical* (at least in the role)
    - some with the role of local **concentrator** that can be activated when needed, the so called “super peers”
  - Example:
    - Fast Track
    - Emule: with the servers for boot
- Super peers
  - facilitating the starting/booting of the peer network,
  - recovering the list of closer peers
  - May create a restricted community around which the content is shared
    - marginally connected with others communities



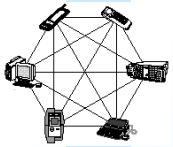
Hierarchical/Hybrid P2P Network





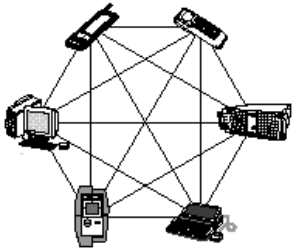
Main Functionalities of Nodes

- **Nodo Centrale di Boot:**
  - NodeList GetList(): to provide list of supernodes level 1
  - AddNode(node): to add a supernode of level 1 to the list
  - DelNode(node): to remove a supernode of level 1 to the list, performed by missing a ping for a while
  - Bool Alive(): to verify if the node is alive
- **Level 1 Node :**
  - NodeList GetList(): to provide list of supernodes level 2
  - Alive(), AddNode(node), DelNode(node), as above
  - Result PassQuery(query): to pass a received query to lower level nodes in its Node2List
  - Result RedirectQuery(query): to pass a received query to nodes of its level: Node1List
- **Level 2 Node: are almost all notifications**
  - It does not receive any command from other nodes on the list
  - Result Query(query): another node is making a query
  - Data GetFileSegment(GUID): to get a file segment
  - Alive(), etc.



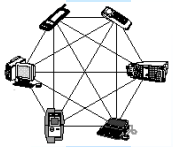
KaZaA example

- It is a **semicentralized P2P solution**
- **Super-peers maintain** info/DB with:
  - file identifiers, their children are sharing
  - metadata (file name, size, contentHash, descriptors)
  - IP addresses of children
- **peers frequently exchange list of super-peers**
  - Peer clients maintains list of 200 super-peers
  - Super-Peers maintain a list of thousands of SPs
- All of the signaling traffic between peers is encrypted
  - Lists, Metadata upload, Queries and replies
- File transfer among nodes is not encrypted
- TCP is used for both file transfer and other communications



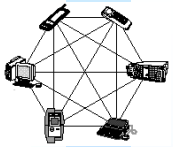
Sistemi P2P

- Aspetti Generali, Applicazioni
- Requirements
- Architecture P2P e caratteristiche
- Ricerche e download multisorgente, BTorrent
- Reti P2P in Overlay
- Esempi: Skype, P2P
- Esempio: P2P distributed trust



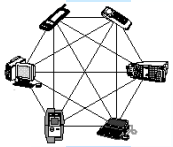
Download Multisorgente

- File diviso in Parti di dimensioni ragionevoli per la rete, qualche Kbyte o decina di Kbyte:
  - F1: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
- Nodi hanno delle parti nella loro memoria cache:
  - N1: 1, 3, 5, 7, 8
  - N2: 2, 4, 5, 7, 8, 10
  - N3: 5, 6, 2, 9
  - Etc..
- Alcuni nodi possono anche averle tutte, cioe' il file completo
- Un nodo puo' scaricare parti diverse da nodi diversi anche allo stesso tempo sfruttando in questo modo un parallelismo
  - P.es.: N3 puo scaricare 4 e 10 da N2, ed 3, 1, 8 da N1



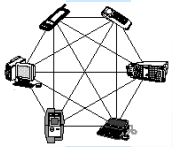
Download Multisorgente

- **Politiche per scaricare le parti/file dai nodi:**
  - Il Nodo permette lo scarico in base ad una coda di richieste
    - Il nodo che chiede viene messo in coda, quando quelli prima hanno avuto almeno una parte vengono messi in fondo alla coda
    - Il nodo può salire nella coda se ha da dare delle parti anche lui all'altro nodo, per esempio
  - Il Nodo ha una limitazione
    - sul numero di scaricamenti contemporanei
    - sulla banda sfruttata in uscita e/o ingresso
  - Un Nodo può acquisire un credito (uno score/voto) in base al suo comportamento nel lasciare scaricare file o nel permettere in uscita una banda larga.
    - In base a questo credito potrebbe/dovrebbe avere delle facilitazioni/score in caso di richieste, per scalare delle posizioni nelle code, etc.



bitTorrent

- **Programma di file sharing**
  - Principalmente P2P, ma con seme iniziale su semplici pagine WEB, .torrent file
  - Open Source
  - Soluzioni in vari linguaggi, C++, Java, etc.
- prestazioni migliori per file di grosse dimensioni
- L'ipotesi e', come per la maggior parte dei sistemi i P2P:
  - che quando uno ha un file anche intero/completo lo continui a condividere con gli altri e non lo tolga dalla directory che contiene i file visibili per gli altri



bitTorrent

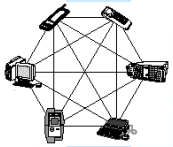
- **Idea:**
  - Un file *.torrent* contiene informazioni su come prendere le parti del file DATI, dove prenderle
    - ➔ chi sono i nodi che hanno porzioni di quel file
  - Il file DATI viene diviso in parti e queste in segmenti
  - Il primo pezzo che viene scaricato e' casuale, i successivi vengono scelti in modo da dare precedenza al più raro, in modo che la sua rarità si attenui visto che viene copiato su di un altro nodo.

A ha 1,4,5,7,8, e metà di 3

B ha 2,3,4,5, e metà di 6

C ha 2,5,6,7 e metà di 8

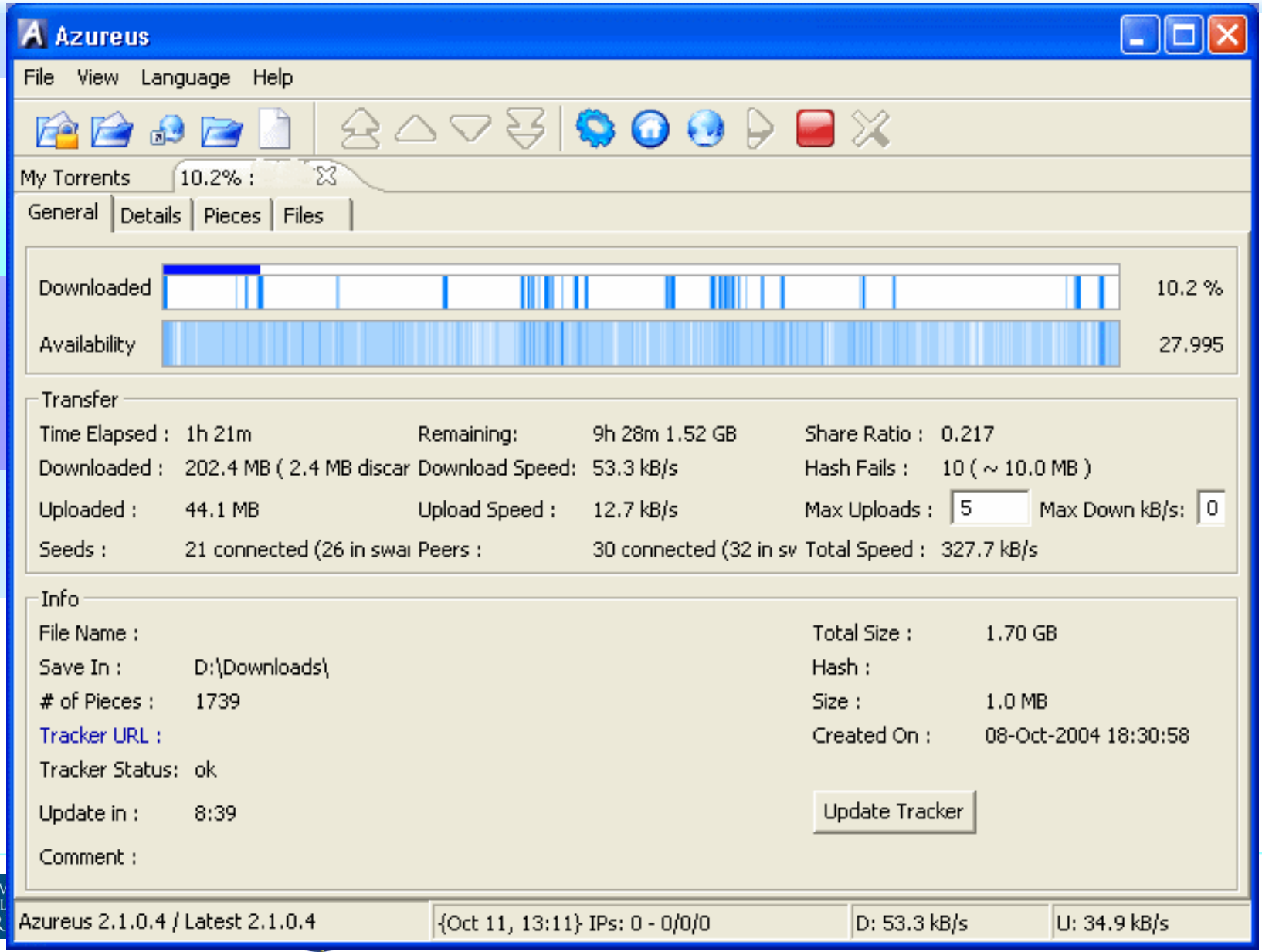
- Quando A finisce con la parte 3, richiede una nuova parte.
- Il programma analizza lo stato generale e trova i pezzi 1 e 6 che sono i più rari.
- Fra questi, A ha bisogno solo della parte 6, così A inizia a scaricare 6
- poi passa agli altri segmenti



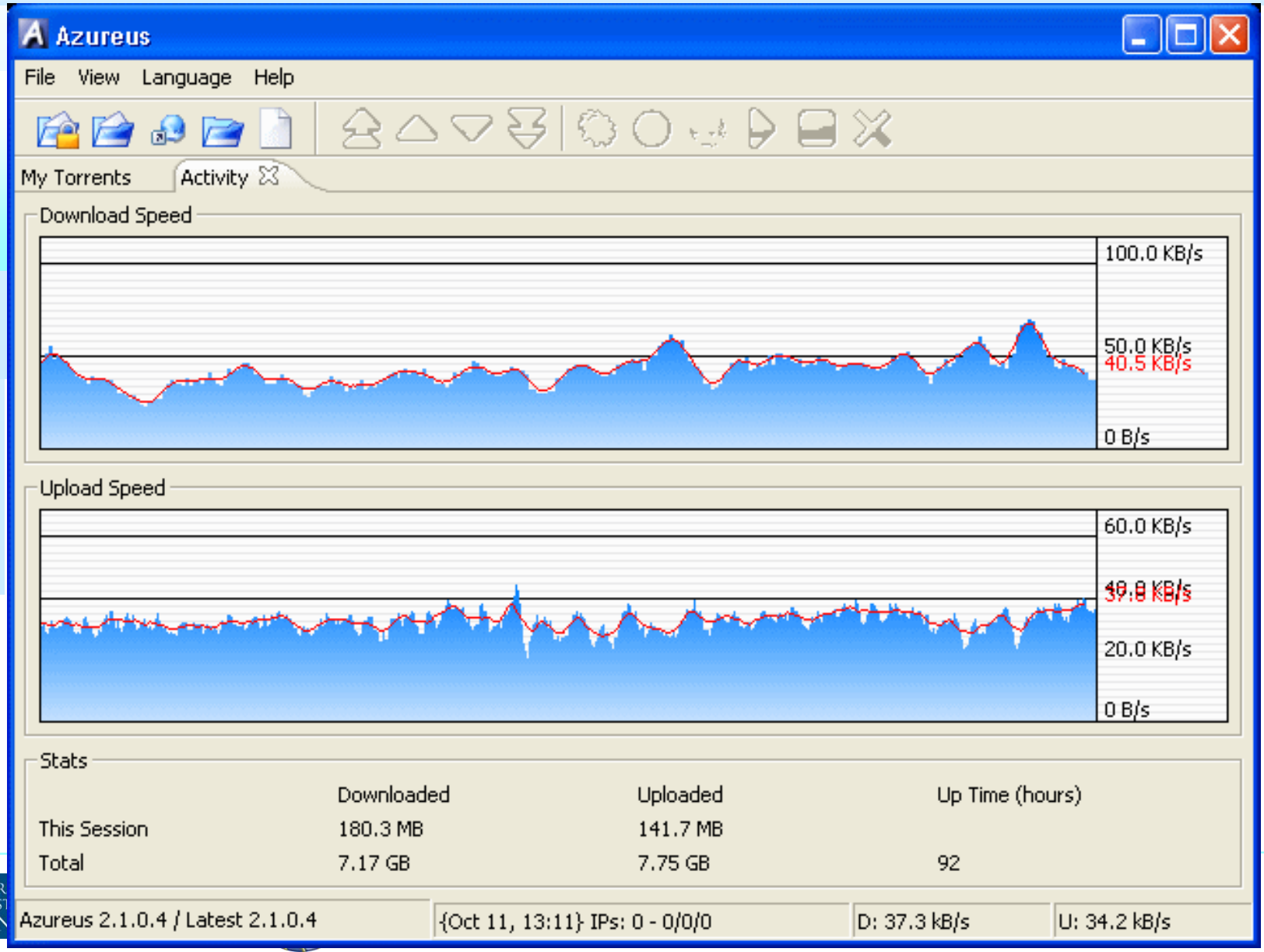
Il tracker in bitTorrent

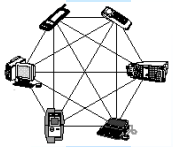
- Quando si effettua una query, la risposta e' una lista di file e per ognuno di questi un file *.torrent*
- Il file *.torrent* contiene informazioni su chi ha i segmenti del file, eventuali duplicazioni, etc.
- Il nodo contatta gli altri peer e parte con lo scarico in base alla strategia vista
- Archivi/tracker diversi possono avere file *.torrent* diversi e questi possono o meno essere manutenuti aggiornati con le informazioni su chi ha il file in questione
- Un client può essere connesso a uno o più *tracker* per la ricerca dei file *.torrent*

Azureus: Monitoraggio dello stato



Azureus: Andamento del download/upload





Azureus: Mappa delle parti

Azureus

File Trasferimenti Visualizza Strumenti Plugins Aiuto

Miei Torrents | Mio tracker | 13,6% : | x1...

Generale | Seeds/Fonti | Swarm | Parti | Files | Impostazioni | Console

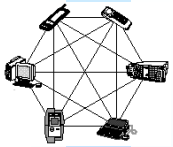
Ip	Client	T	Parti Disponibili	%	Velocità...	Velocità...	Stato
151.32.188.6	BitComet 0.70	R		47,1%	7,1 kB/s	17,2 kB/s	Piename...
80.116.146.218	BitComet 0.70	L		13,3%	2,0 kB/s	4,1 kB/s	Piename...
200.141.184.74	µTorrent 1.6.0	R		60,0%	1,1 kB/s	0 B/s	Piename...

Mappa Delle Parti | Console

µTorrent 1.6.0; 200.141.184.74; 60,0%

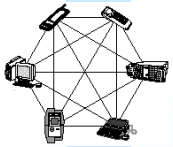
Entrambi Lo Possiedete | Peer Possiede; Tu No | Lo Possiedi; Il Peer No | Nessuno Lo Possiede | In Traferimento | Prossima Richiesta | Disponibilita

Azureus 2.4.0.2 | Ratio | NAT OK | 740.092 Utenti | {lug 26, 14:21} IPs: 0 - 0/0/0 | 24,1 kB/s | [40K] 39,3 kB/s



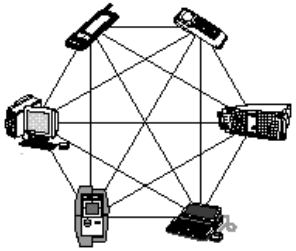
BitTorrent Terminology

- **Choked**: a peer to whom the client refuses to send file pieces.
- **interested** a downloader who wishes to obtain pieces of a file the client has.
- **leech** a peer who has a negative effect on the swarm by having a very poor share ratio - in other words, downloading much more than they upload.
- **peer** one instance of a BitTorrent client running on a computer on the Internet to which other clients connect and transfer data.
- **seeder** a peer that has a complete copy of the torrent and still offers it for upload.
- Nodes of the BT are the peers and seeders



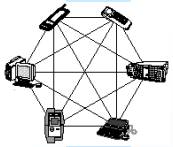
BitTorrent Terminology

- **superseed** When a file is new, much time can be wasted because the seeding client might send the same file piece to many different peers, other pieces have not yet been downloaded at all.
  - Some clients, like ABC, Azureus, BitTornado, TorrentStorm, and µTorrent have a "superseed" mode,
  - they try to only send out pieces that have never been sent out before, making the initial propagation of the file much faster.
- **swarm** all peers (including seeders) sharing a torrent are called a swarm.
  - For example, six ordinary peers and two seeders make a swarm of eight.



Sistemi P2P

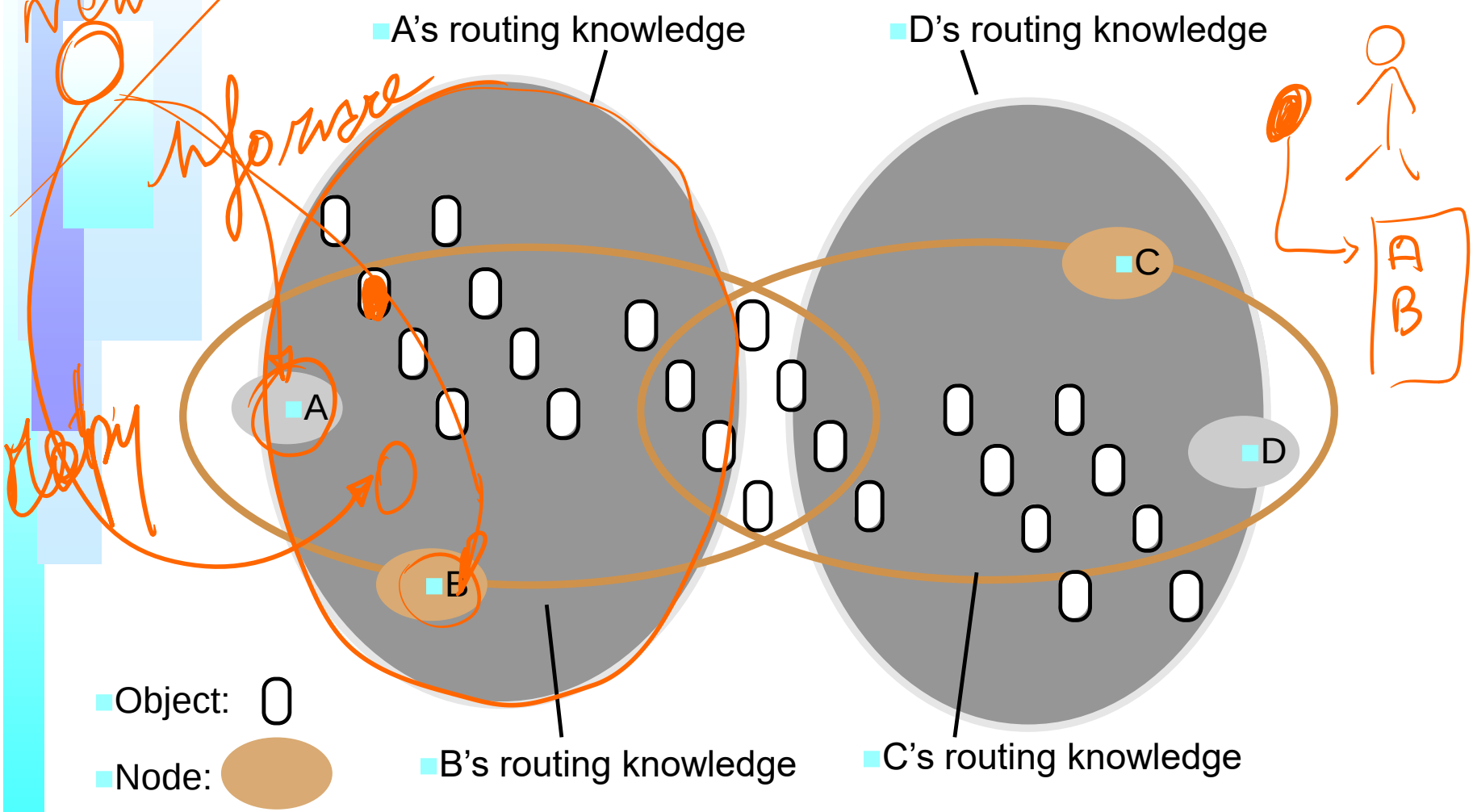
- Aspetti Generali, Applicazioni
- Requirements
- Architecture P2P e caratteristiche
- Ricerche e download multisorgente, BTorrent
- Reti P2P in Overlay
- Esempi: Skype, P2P
- Esempio: P2P distributed trust

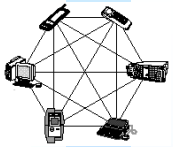


Rounting Overlay

- Soddisfare tutti i requisiti precedenti e' molto complesso
- RO garantisce che ogni nodo può accedere ad ogni oggetto instradando la richiesta al fine di far raggiungere il nodo dove si trova la risorsa/info (tramite una sequenza di nodi)
  - L'oggetto può essere spostato in altri nodi senza coinvolgimento degli utenti
  - Si crea una catena di riferimenti
  - Usato in molti casi, vedasi: Skype, P2PTV, etc.
- Sistemi P2P usualmente replicano la risorsa
  - In questo caso, l'algoritmo di RO deve tenere conto di dove sono le repliche e può facilitare la consegna fornendo a fronte delle richieste/query il nodo più vicino

Distribution of information in a routing overlay

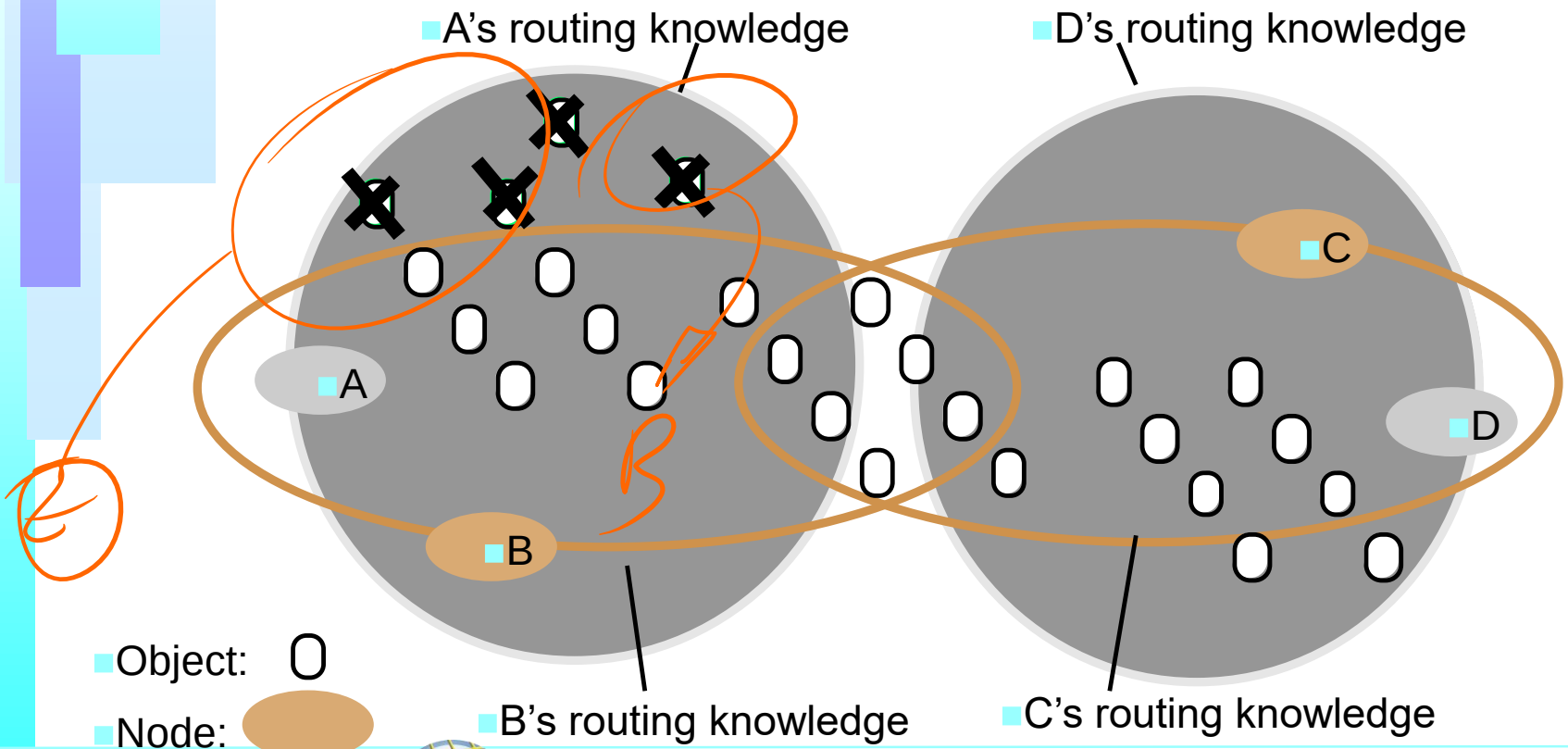




Rounting Overlay

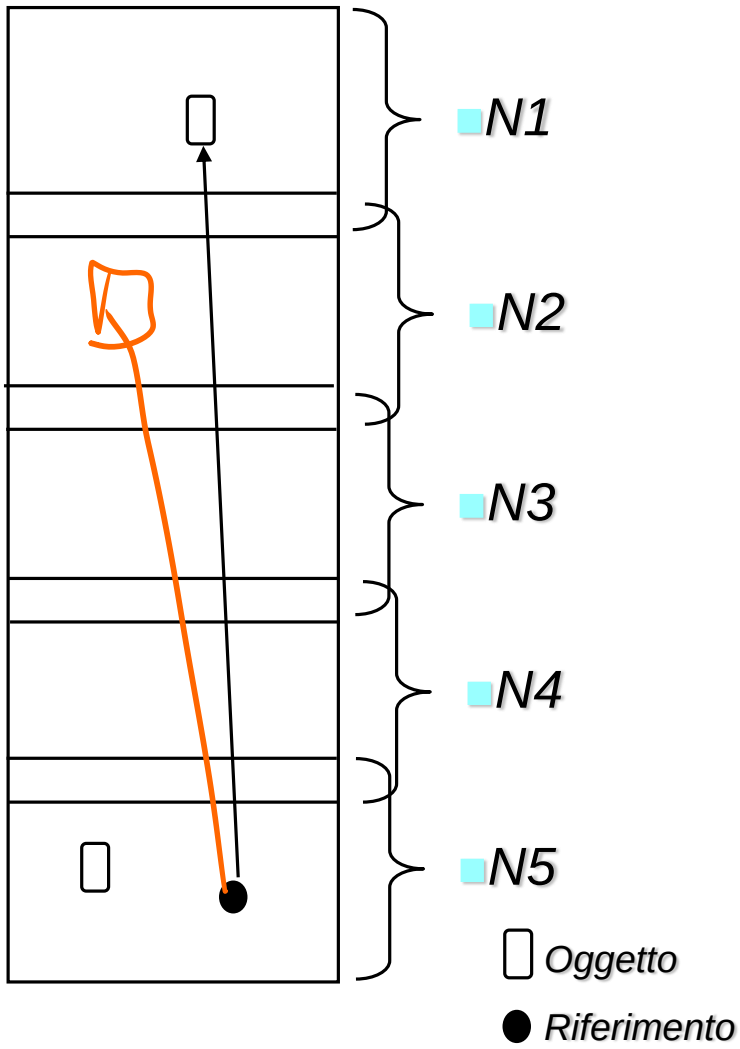
- Le richieste possono essere effettuate tramite un GUID (global unified ID)
  - La richiesta fatta al RO produce in risposta il/un nodo che ha la risorsa/info.
- L'algoritmo di RO deve anche:
  - Pubblicare/Rendere-noto a tutti i nodi le eventuali nuove pubblicazioni di GUID
  - **PRO**: Poter cancellare da tutti i nodi gli oggetti e pertanto la loro GUID che e' stata rimossa
  - **PRO**: Rendere aggiornati i nuovi nodi con la lista dei GUID dandogli alcune delle responsabilita', la gestione del segmento di conoscenza che loro rappresentano
  - **CONTRO**: Al momento in cui un nodo lascia la rete deve ridistribuire le responsabilita'/(la conoscenza) ai nodi che rimangono. In modo da non creare delle falle.

Al momento in cui un nodo lascia deve ridistribuire le responsabilità/(la conoscenza) ai nodi che rimangono. In modo da non creare delle falle. Se A va via, gli oggetti X devono essere presi in carico da B o da altri, altrimenti vengono persi.



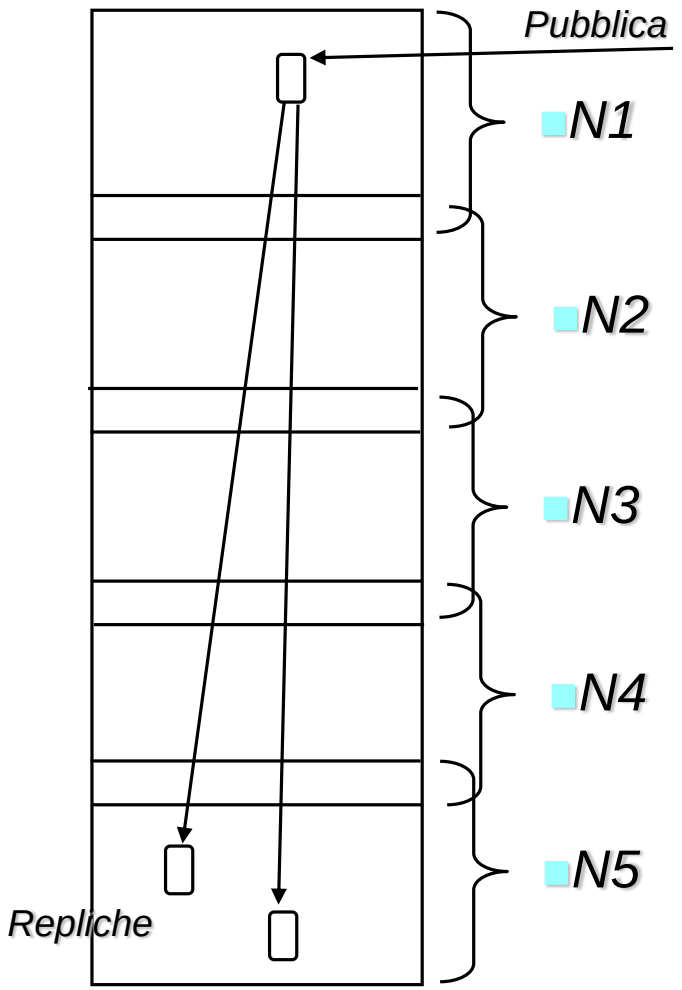
GUID and DHT

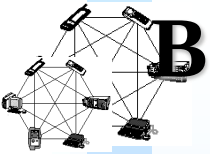
- GUID puo' essere calcolato tramite:
  - HASH function delle info
- Pertanto il problema e' simile ad avere una tabella Hash distribuita: Distributed Hash Table, DHT.
- Si veda a destra una semplificazione



Routing Overlay e repliche

- un algoritmo random decide dove mettere l'oggetto e le sue repliche in modo da assicurare la loro accessibilita'
- Il numero di repliche puo' essere variabile. Se l'alg. Random identifica ancora lo stesso nodo deve essere ricalcolata un nuova posizione
- La posizione dipende dal valore di GUID dell'oggetto
- Un oggetto con GUID x (e.g., 5) viene posto in nodi che hanno GUID prossimi/vicini in modo da massimizzare la probabilità di trovarlo in fase di ricerca.





Basic programming interface for a distributed hash table (DHT) as implemented by the PAST API over Pastry

put(GUID, data)

The *data* is stored in replicas at all nodes responsible for the object identified by *GUID*.

remove(GUID)

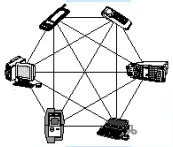
Deletes all references to *GUID* and the associated data.

Solo accedendo a quelli che coprono tale conoscenza.

Pertanto se vi sono delle repliche prodotte da utenti, possono essere o meno cancellate se non si operano particolari accorgimenti. Comunque non sono piu' recuperabili da altre operazioni di GET pertanto la rete non le considera piu'

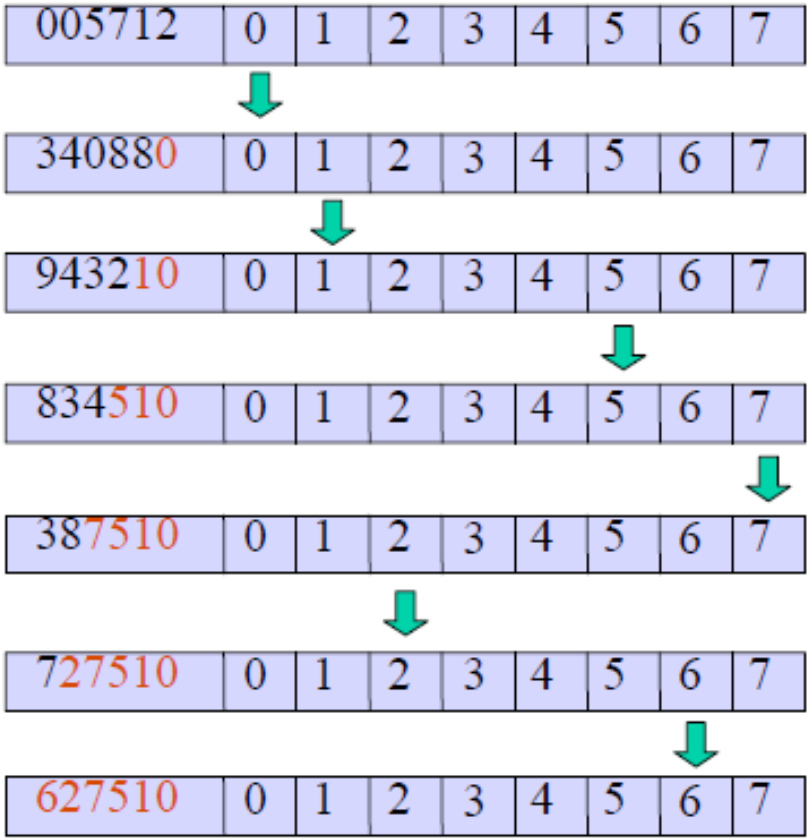
value = get(GUID)

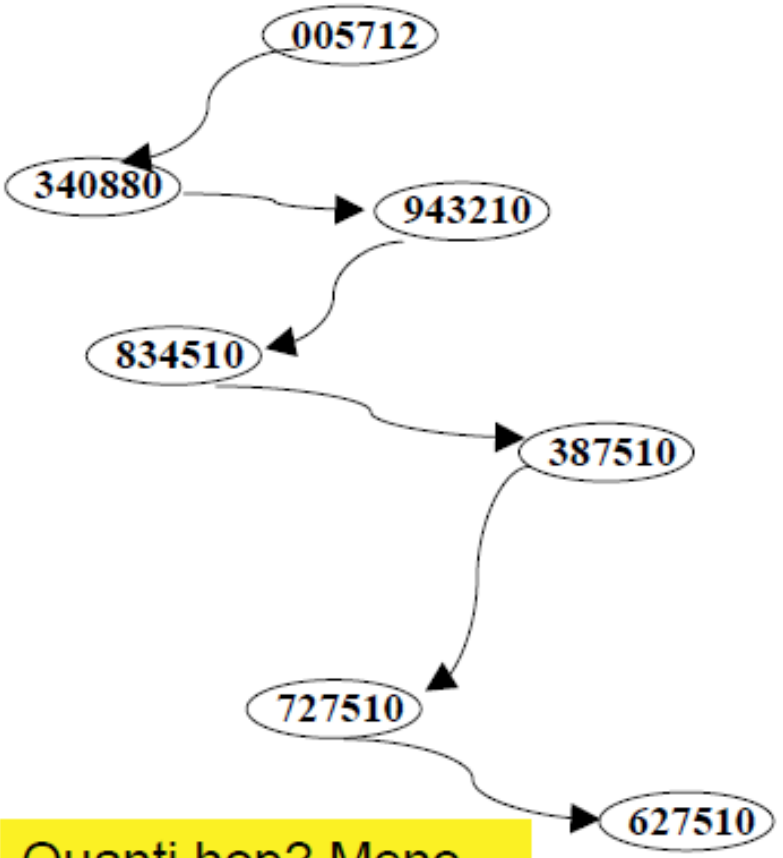
The data associated with *GUID* is retrieved from one of the nodes responsible for it. Quelli che coprono quella conocenza



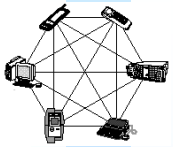
Pastry (esempio preso da Cardellini) (3 bit per ogni cifra)

Consideriamo un namespace di 2^{18}, 005712 → 627510



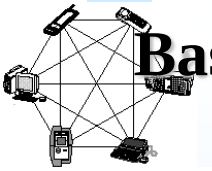


Quanti hop? Meno di $\lceil \log_{2^b} N \rceil$



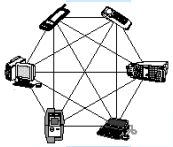
Distributed Object Location and Routing

- **DOLR**: e' un modello di RO leggermente migliorato
- **Idea di base:**
  - Gli oggetti sono disposti dove si vuole, sono i riferimenti che vengono disposti sulla base del GUID
  - DOLR ha il compito di definire un mapping fra gli indirizzi dei nodi che contengono repliche e gli oggetti (con i loro GUID)
  - Gli oggetti sono memorizzati con lo stesso GUID in nodi diversi, questi sono repliche
  - RO ha la responsabilità di instradare le richieste verso il nodo più vicino al richiedente
- **Posizione degli oggetti:**
  - Le repliche sono poste senza considerare la vicinanza del valore di GUID secondo delle politiche: e.g., random
  - Ogni replica deve essere notificata al DOLR tramite Publish()



Basic programming interface for distributed object location and routing (DOLR) as implemented by Tapestry

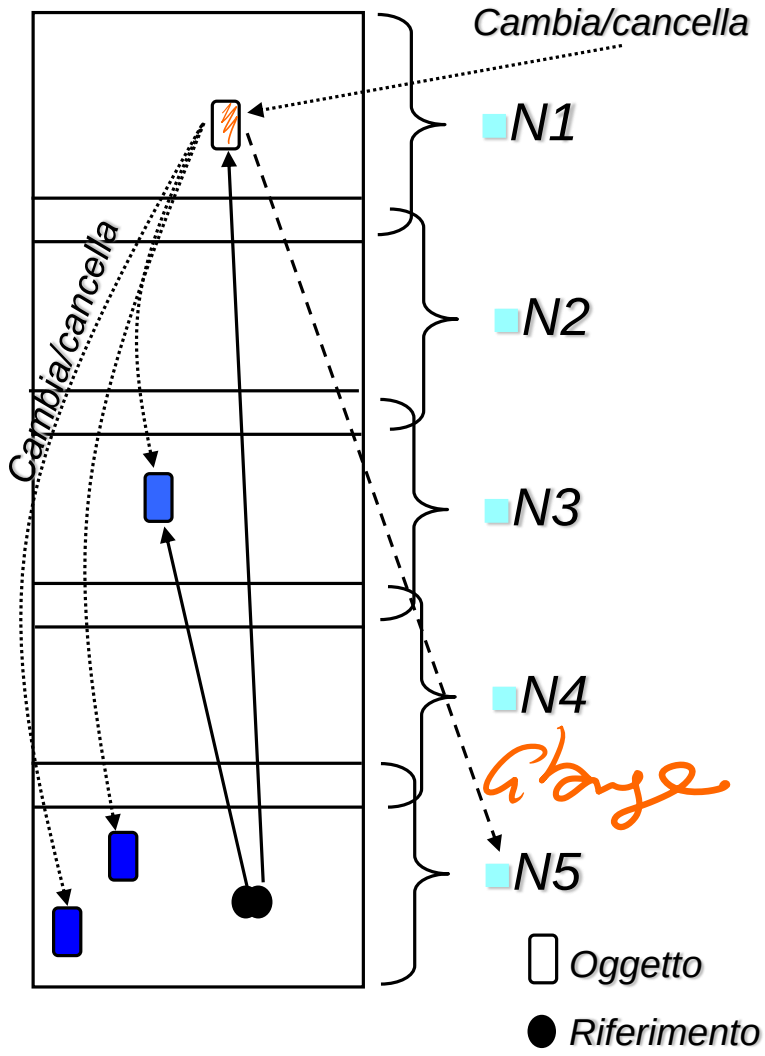
- ***publish(GUID)***
  - *GUID* can be computed from the object (or some part of it, e.g. its name). This function makes the node performing a *publish* operation the host for the object corresponding to *GUID*.
- ***unpublish(GUID)***
  - Makes the object corresponding to *GUID* inaccessible.
- ***sendToObj(msg, GUID, [n])***
  - Following the object-oriented paradigm, an invocation message is sent to an object in order to access it. This might be a request to open a TCP connection for data transfer or to return a message containing all or part of the object's state. The final optional parameter *[n],* if present, requests the delivery of the same message to n replicas of the object.

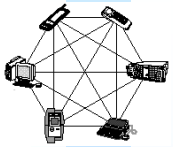


Cambiamenti sui file

- Se un certo oggetto e' stato cambiato/cancellato
  - Notifcation to who:
    - has performed the download for that object in the past or
    - is managing replica for that object
    - has references for that object
  - The object has to be reloaded, replicated again, substituting the old one, not very nice privacy problems

- **If the object is**
  - not replicated the change is immediate
    - Who has downloaded has to be informed as well
  - replicated:
    - Make a query to know where the object is replicated
    - removing/deleting the old versions
    - Put/publish the new one





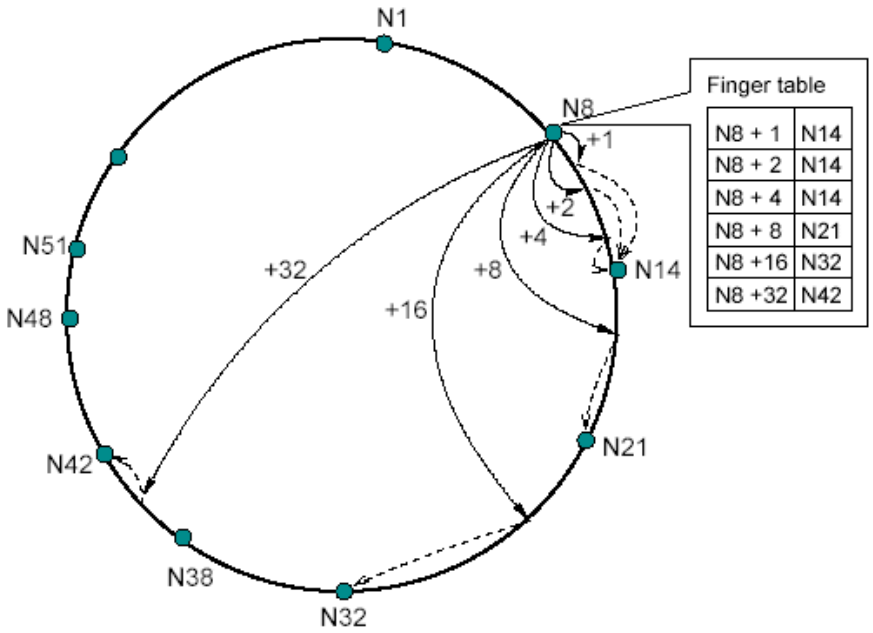
Come instradare

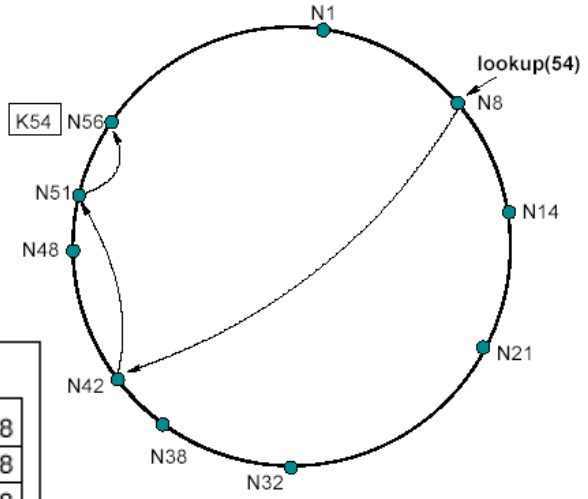
- **Pastry e Tapestry** usano il Prefix Routing per determinare il percorso per l'instradamento per la consegna dei messaggi/pacchetti/richieste indirizzate ad un certo GUID.

- Idea di base:
  - Modelli basati sulla disposizione dei nodi in base ad una gerarchia come per esempio in routing IP packets:
    - ogni byte identifica 256 possibili figli, ogni figlio 256 figli, etc.. IP4 has 4 livelli.
    - Segmento il GUID in livelli, etc.
  - Praticamente dei meccanismi che permettono di definire delle distanze massime fra nodi

Criteri per la stima della distanza

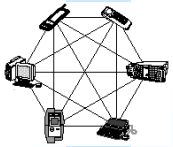
- **CHORD** come distanza usa la differenza fra il GUID del nodo presente e di quello che si cerca.
  - Distanza in un modello Hash uniforme
  - Nodi geograficamente distanti potrebbero trovarsi vicini nello spazio della tabella, questo non e' positivo per ottimizzare i tempi di comunicazione visto che nodi vicini si devono parlare spesso
  - Si basa su un match esatto della stringa di ricerca





Finger table

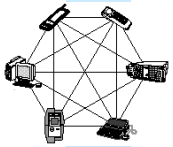
N42+ 1	N48
N42+ 2	N48
N42+ 4	N48
N42+ 8	N51
N42+16	N1
N42+32	N14



Criteri per la stima della distanza

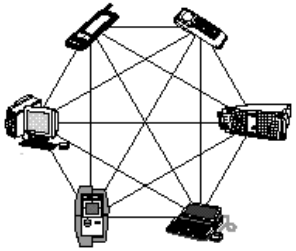
- **CAN**: usa una distanza *d*-dimensionale nell'iperspazio delle possibili posizioni dei nodi
  - Dimensioni per esempio possono essere: IP (geografico), location (nationality), language, fuso orario, codice postale, hash, etc.

- **Kademlia**: usa lo XOR sulla coppia di GUID come distanza fra i nodi
  - Anche questo puo' evere i problemi che si hanno per CHORD.



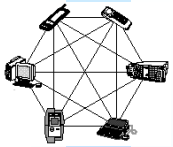
GUID storing

- GUID non hanno senso per gli umani sono semplici codici binari/esadecimali/ottali “lunghi”, non hanno un significato diretto alla loro lettura
- GUID puo’ essere calcolato sulla base dei dati che compongono l’oggetto e/o le informazioni del nodo, per esempio sulla base delle keyword che lo descrivono, il file name, etc.
- potrebbero essere anche ottenuti tramite lo standard UUID che pero’ produce un valore assoluto non ricostruibile dai dati dell’oggetto e dovrebbe essere dato da un ente superparte, che oltre che generare l’ID verifica di non avere dei duplicati



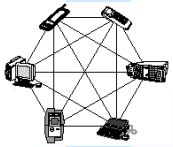
Sistemi P2P

- Aspetti Generali, Applicazioni
- Requirements
- Architecture P2P e caratteristiche
- Ricerche e download multisorgente, BTorrent
- Reti P2P in Overlay
- Esempi: Skype
- Esempio: P2P distributed trust



Skype P2P architecture

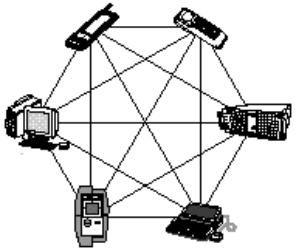
- Skype is a P2P application for VOIP, Voice Over IP
- Skype has a proprietary protocol,
  - the P2P is similar to KaZaA
  - AES 256 is used to protect the channel
- There are three types of nodes in the P2P network:
  - Ordinary-peers (OP, the client), Super-peers, Central login server
  - The boot of OP is performed on a Super Peer (SP), and ask to the Central server to perform the authentication
- For user search
  - OP send the user name to SP which provide 4 IP addresses, if it is not there, with another request to SP obtain 8 peers, etc…



Some info on Skype

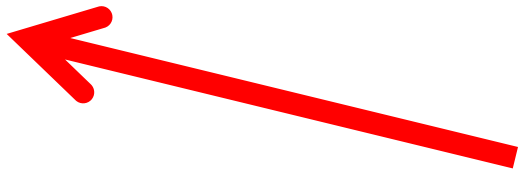
- Skype usa il P2P per implementare la directory distribuita degli utenti
  - DHT – Chord algorithm
  - Costo della ricerca: O(log N)

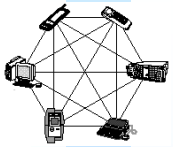
- For user search
  - OP send the user name to SP which provide 4 IP addresses, if it is not there, with another request to Sp obtain 8 peers, etc…



Sistemi P2P

- Aspetti Generali, Applicazioni
- Requirements
- Architecture P2P e caratteristiche
- Ricerche e download multisorgente, BTorrent
- Reti P2P in Overlay
- Esempi: Skype, P2P per il B2B, basata su BTorrent
- Esempio: P2P distributed trust



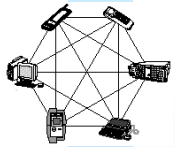


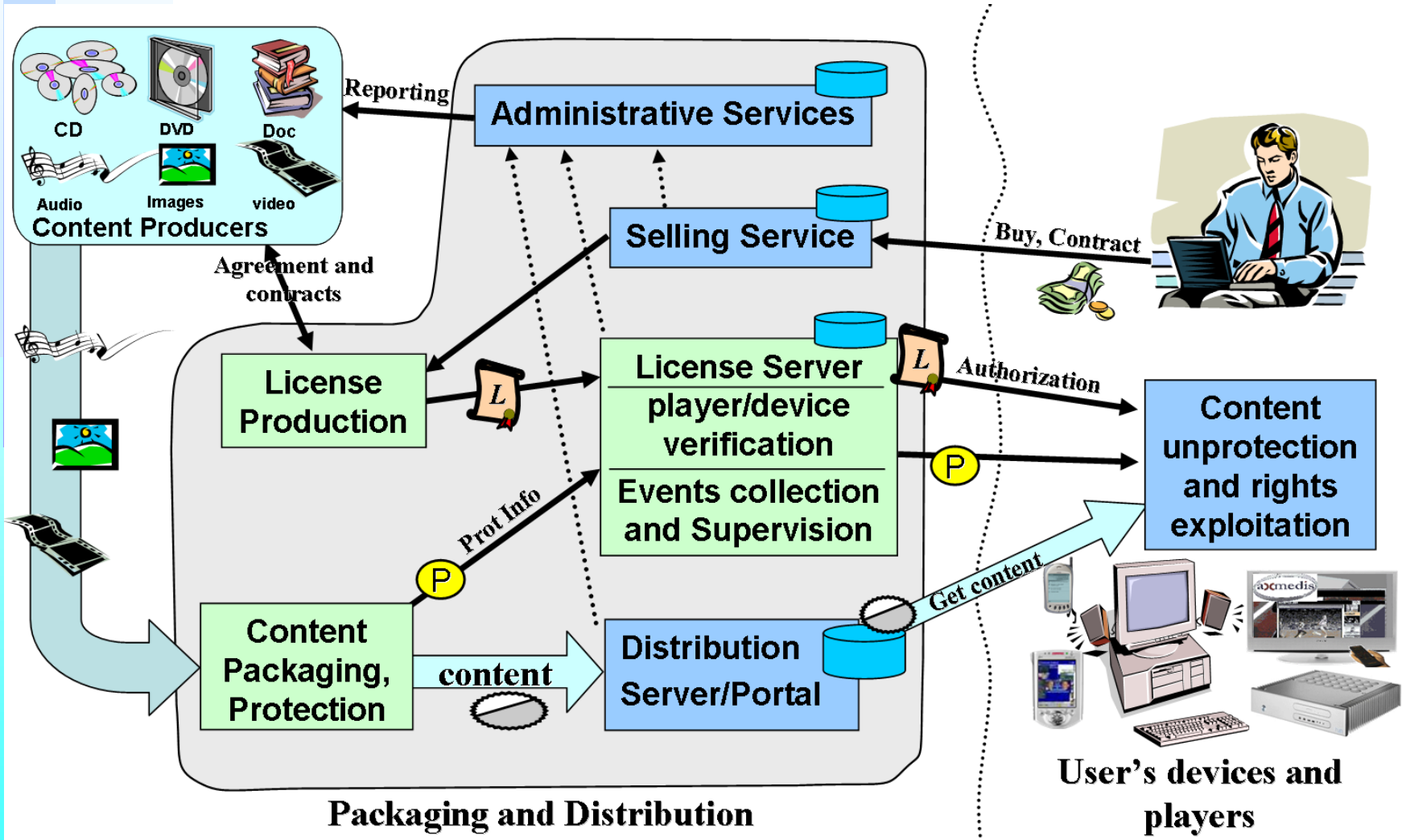
P2P reciprocal Trusting for DRM

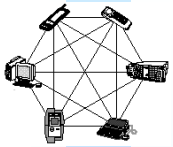
- le prestazioni del DRM AXMEDIS sono fortemente condizionate dal numero di chiamate contemporanee;
  - Ad esempio, in certi casi, ogni 8 richieste di autorizzazione in contemporanea, 2 non hanno esito positivo, a causa dei ritardi nell'elaborazione;
  - il DRM AXMEDIS ha complessità computazionale lineare 1:N (1 server:N client), e.g., 10 alla 12. O(R), O(U*C), un milione di utenti per un milione di content.

Obiettivi:

- **Scalabilità**: definire e sviluppare una soluzione scalabile che consenta di far fronte a carichi elevati sul sistema, decentralizzando le risorse da aggiungere, in modo da poter soddisfare un elevato numero di richieste contemporanee.
- **Sicurezza**: adottare politiche adeguate alla nuova soluzione, mantenendo gli stessi standard di sicurezza



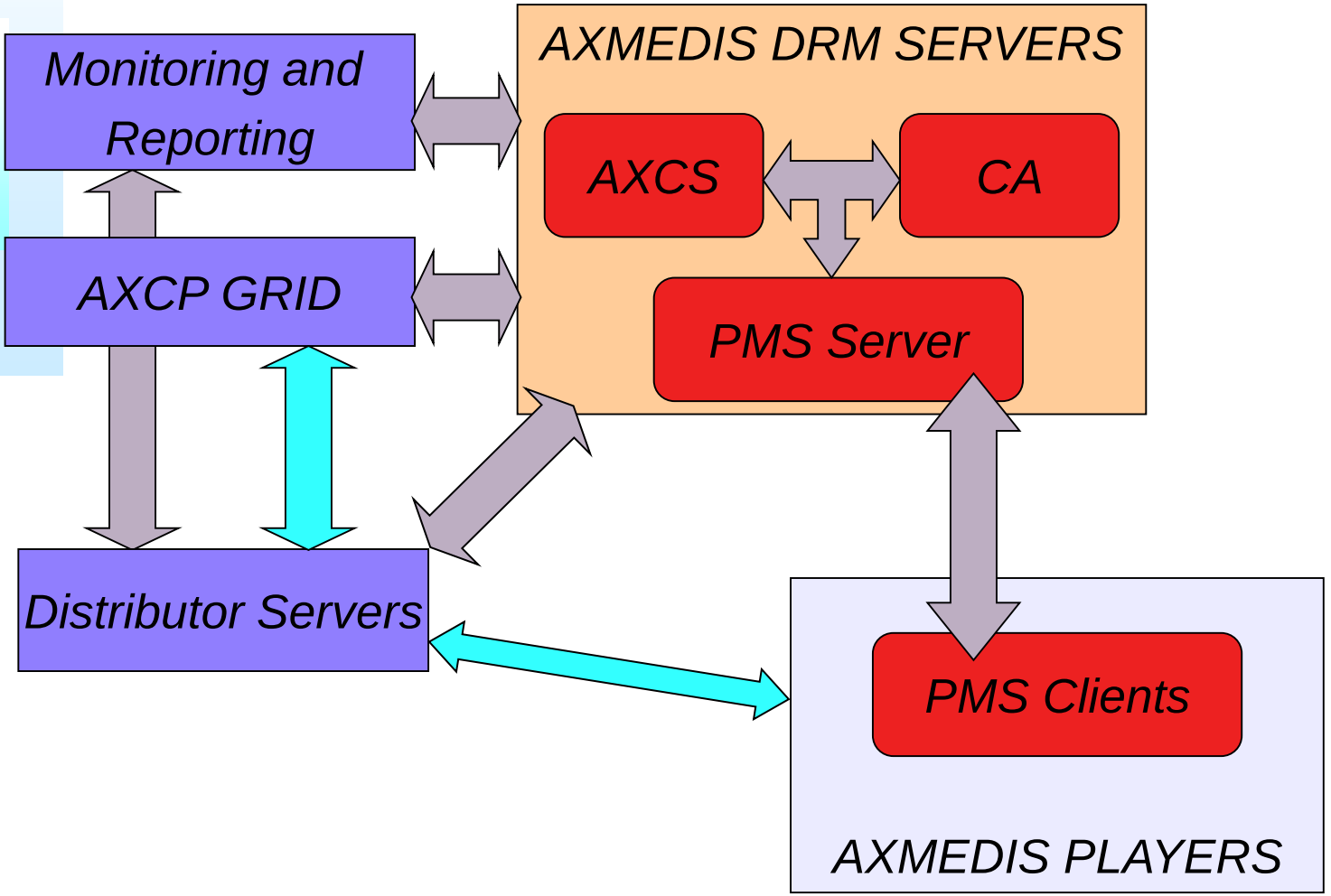

CD
DVD
Doc
Audio
Images
video
Content Producers
Reporting
Administrative Services
Selling Service
Buy, Contract
Agreement and contracts
License Production
L
License Server
player/device verification
Events collection and Supervision
L
Authorization
P
Content unprotection and rights exploitation
Prot Info
P
Content Packaging, Protection
content
Distribution Server/Portal
Get content
Packaging and Distribution
User's devices and players

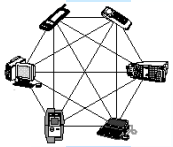


Richiami sul DRM

- **AXMEDIS e'**
  - **un'**infrastruttura distribuita multipiattaforma per la creazione, protezione, distribuzione e consumo delle risorse digitali
- **Digital Rights Management, DRM**:
  - meccanismi di gestione dei diritti di risorse digitali, grazie alla possibilità di rendere protette, identificabili e tracciabili le risorse stesse
- **DRM AXMEDIS** fornisce
  - una serie di strumenti che consentono di
    - certificare utenti e dispositivi,
    - gestire i diritti e
    - controllarne lo sfruttamento da parte degli utenti finali

Architettura DRM AXMEDIS

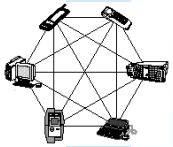




DRM AXMEDIS/MPEG-21

- Procedure fondamentali del DRM:

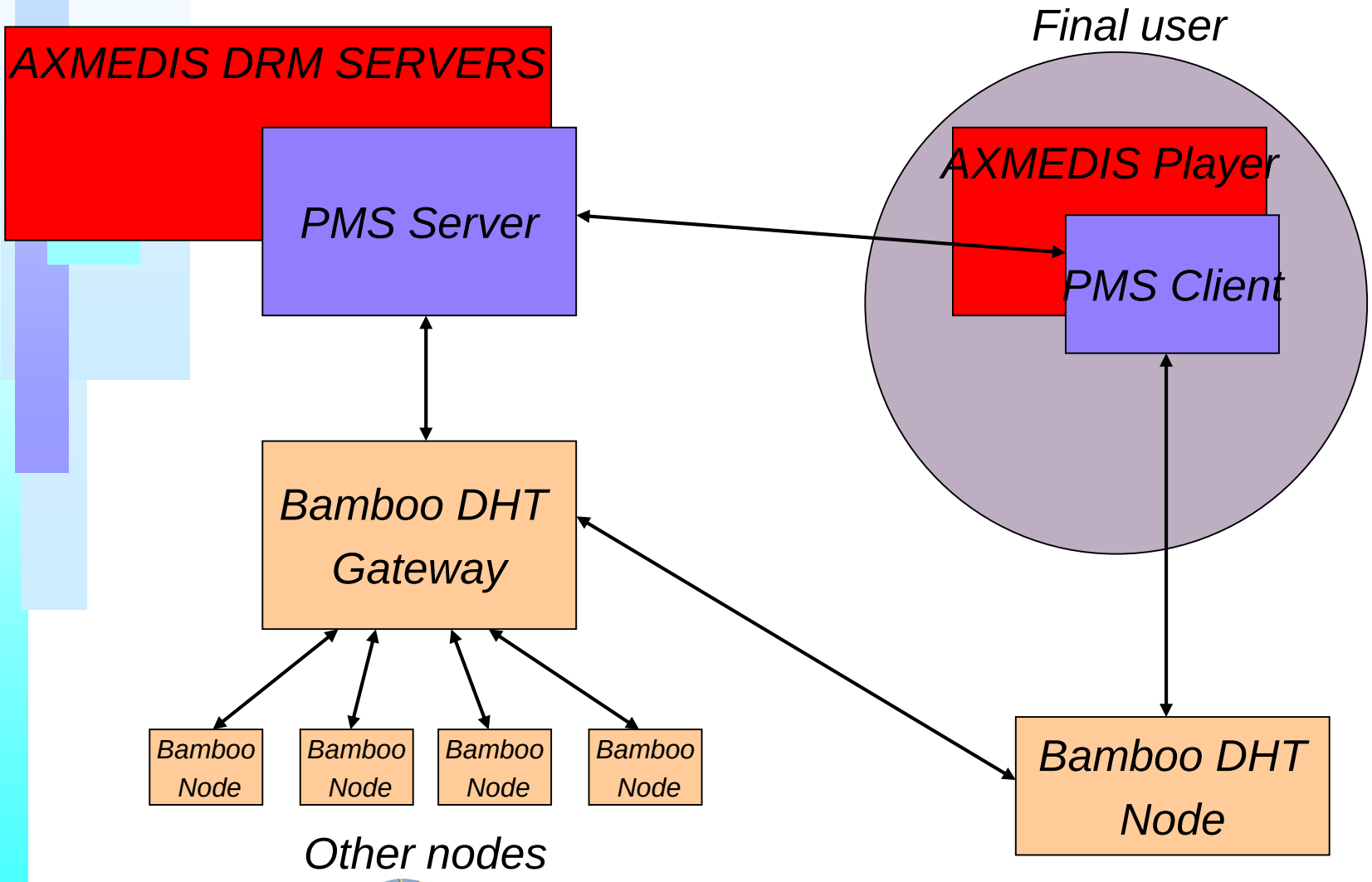
- ***Server side***
  - *registrazione di: utente e tool type*
  - *protezione e registrazione del contenuto*
  - *registrazione della licenza*
- ***Client side***
  - *certificazione del tool: salvataggio dei dati relativi al tool utilizzato*
  - *verifica del tool: controllo sull'integrità*
  - *autorizzazione: concessione del grant all'utente finale*

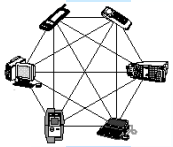


Scelta della struttura decentralizzata

- **Si concretizza la decentralizzazione** scegliendo di delegare l'esecuzione di alcune procedure chiave ad una DHT su rete P2P
  - DHT: Distributed Hash Table:
  - Classe di sistemi distribuiti decentralizzati che partizionano l'appartenenza di un set di chiavi tra i nodi che fanno parte della rete P2P;
  - hanno complessità computazionale logaritmica
- Perché Bamboo DHT:
  - *due insiemi di vicini: leaf set e routing table*
  - *interfacce put/get predefinite: XML-RPC*
  - *gestione del churn*
  - *open source*

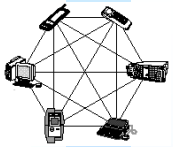
Interazioni tra nodi e DRM





Salvataggio dati sulla DHT

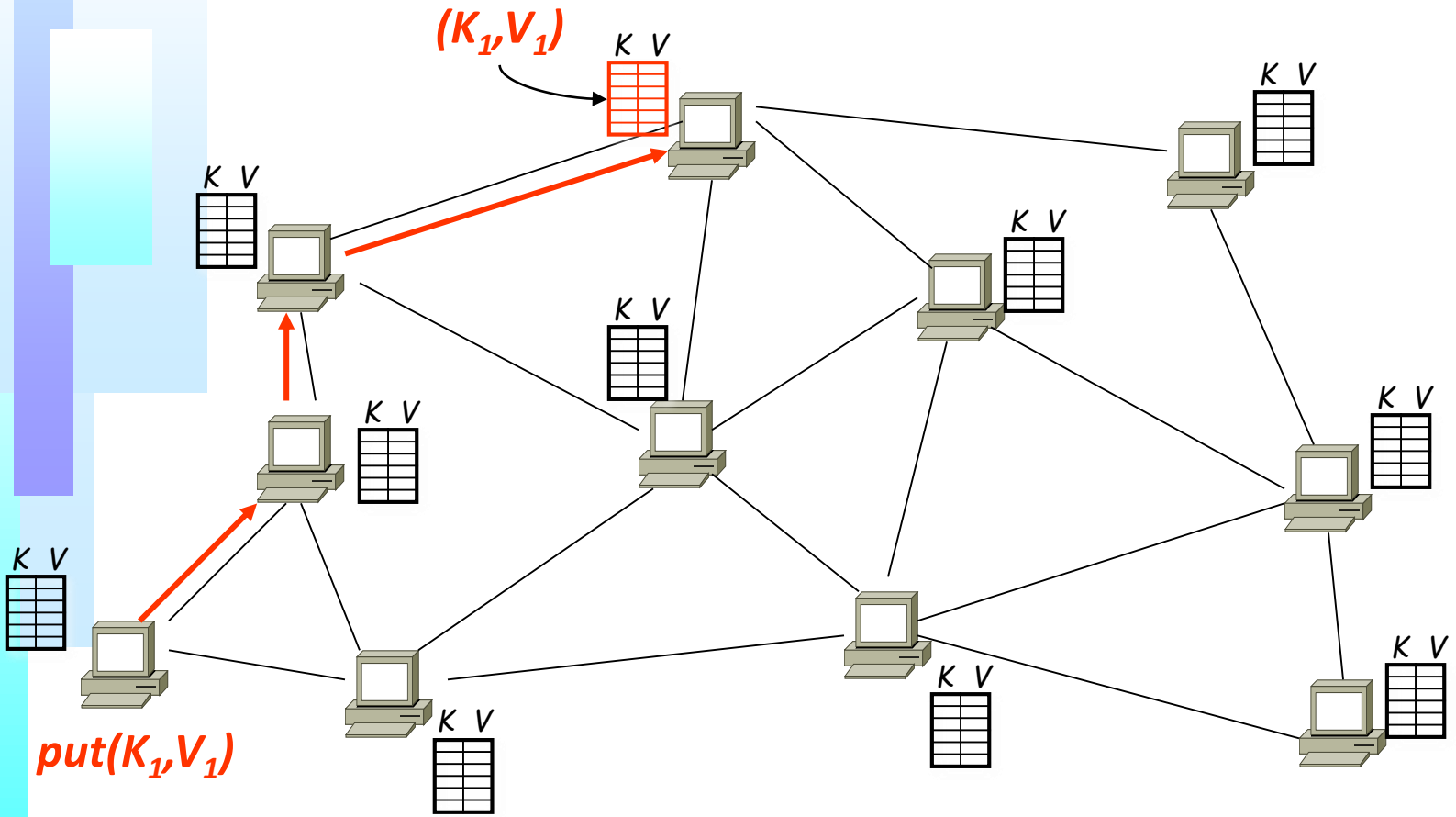
- **avviene** nei casi di
  - certificazione del tool e
  - registrazione della licenza.
- **si effettua** durante
  - la verifica del tool e
  - l'autorizzazione per l'utente finale.
- **Il PMS Server**, completate le rispettive procedure, effettua
  - una connessione ad un nodo Bamboo Gateway per trasmettere le informazioni utili.



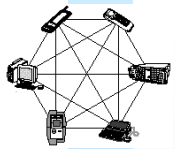
Recupero dati dalla DHT

- **Si effettua, come scelta prioritaria**,
  - ogni volta che si eseguono le procedure di verifica e autorizzazione.
- **Spetta al PMS Client** effettuare un lookup sulla DHT,
  - in base alla chiave specifica per la relativa procedura.
- **Quando viene trovata una corrispondenza sulla rete P2P**,
  - la procedura viene espletata direttamente,
  - senza dover ricorrere ai server di DRM AXMEDIS.

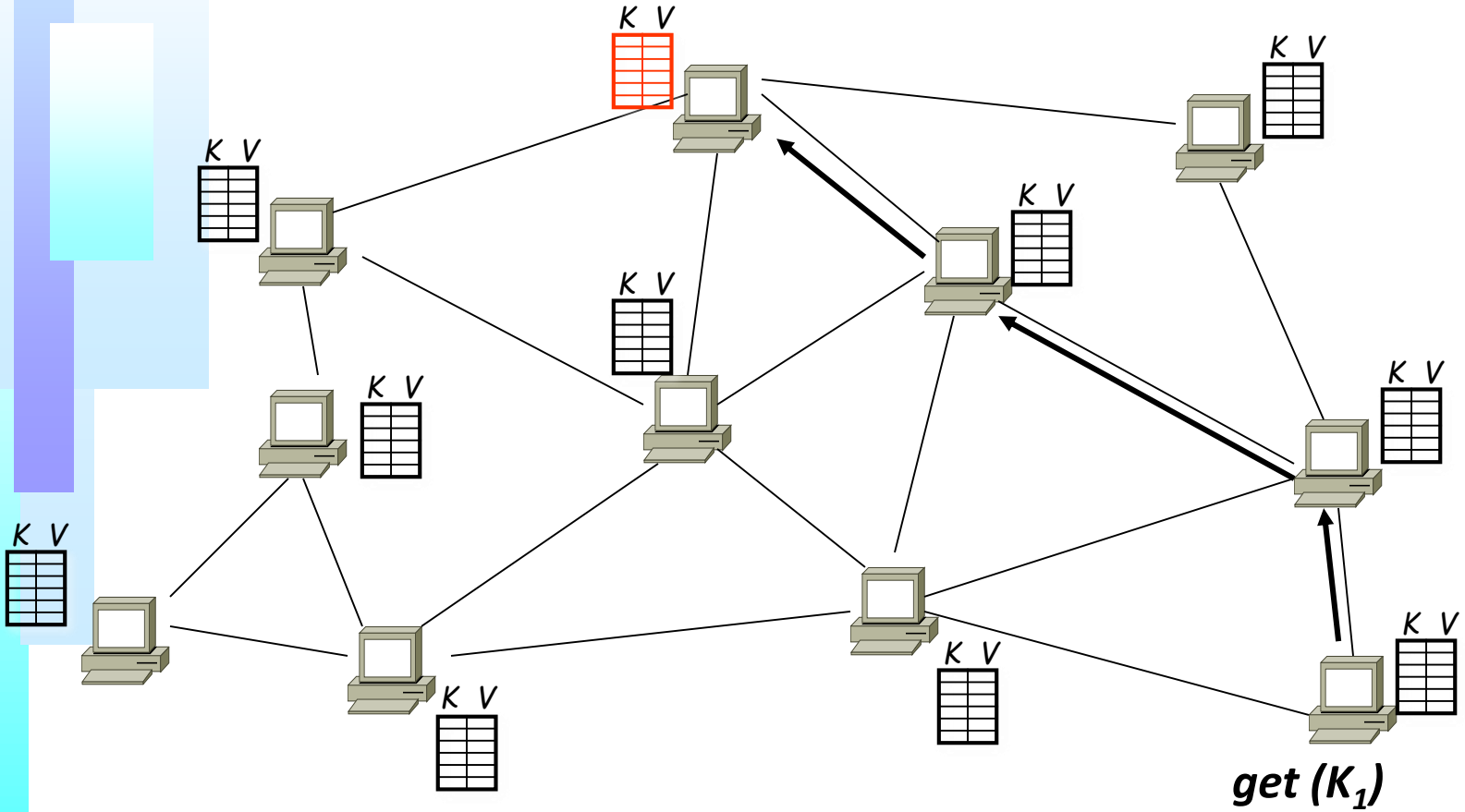
DHT in action: put()



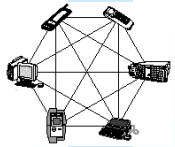
Operation: take key as input; route msgs to node holding key



DHT in action: get()



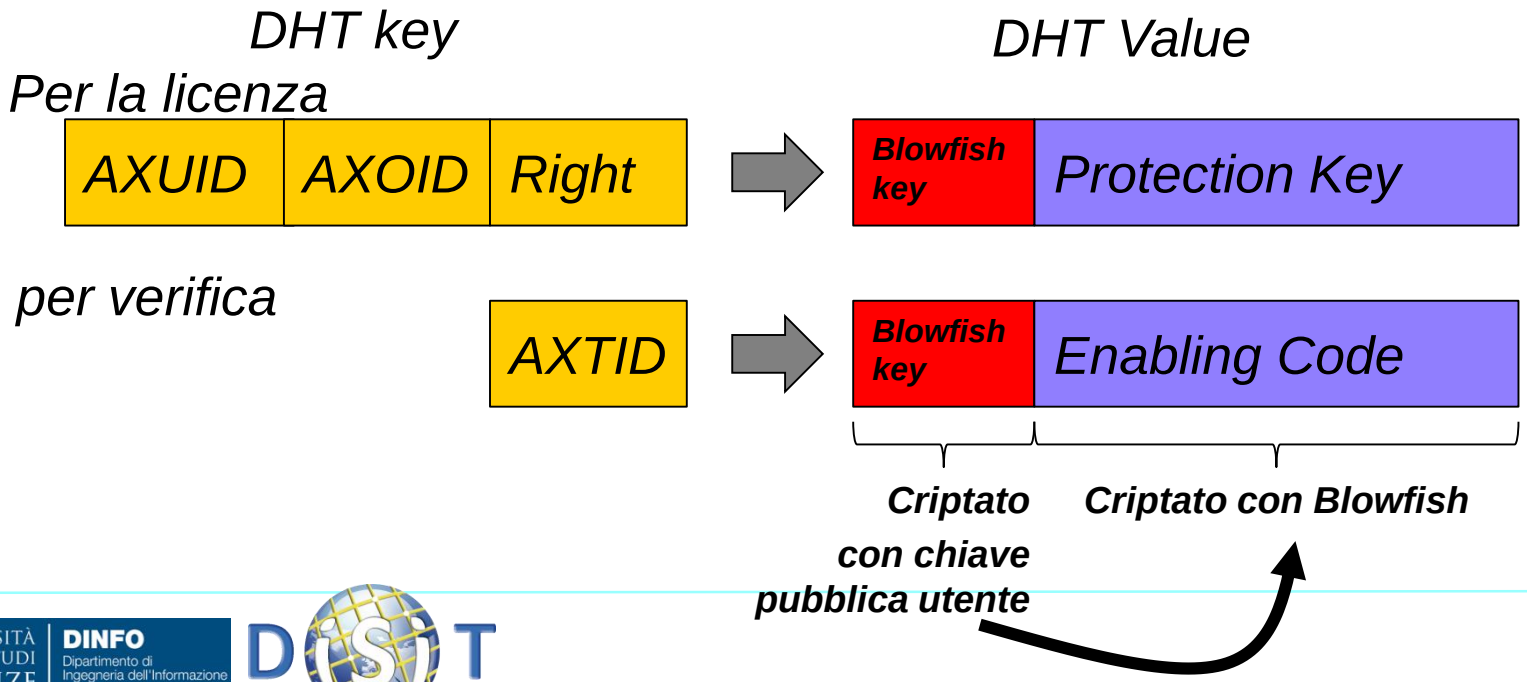
Operation: take key as input; route msgs to node holding key



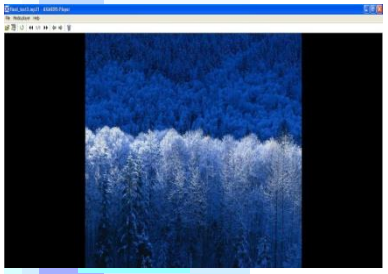
- n keyspace composto da stringhe di 160 bit
  - Al massimo di hanno u numero di hop pari al Log in base due di 160
  - Costo della ricerca: O(log N)
- Se si vuole immettere nella DHT un contenuto caratterizzato dai parametri *filename* e *data*, viene inizialmente calcolato l'hash del filename, ad esempio tramite un algoritmo SHA
- Bamboo è un sistema peer to peer che riassume alcune delle caratteristiche di Chord e Tapestry. È stato scritto in Java da Sean Rhea della UC Berkeley, ma si basa fortemente sui progetti OceanStore e Libasync, ed è open source

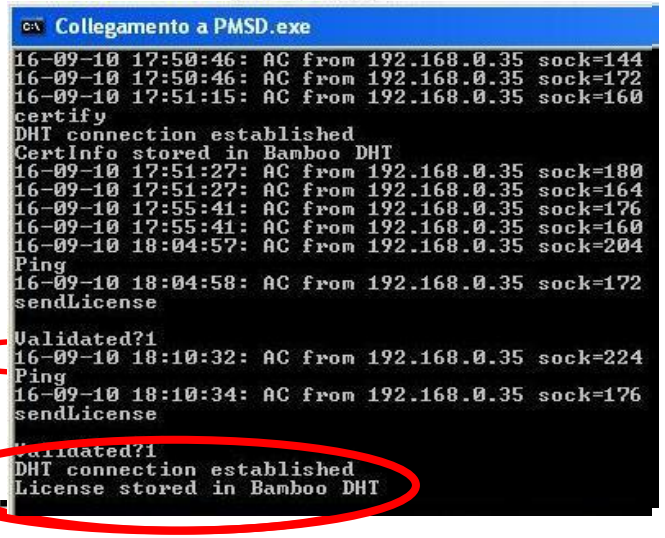
Sicurezza dei dati

- Sicurezza ottenuta tramite meccanismi di crittazione: Blowfish e RSA.
- Si ottiene quindi un doppio sistema di cifratura, che impedisce agli utenti non autorizzati l'utilizzo delle informazioni riservate.



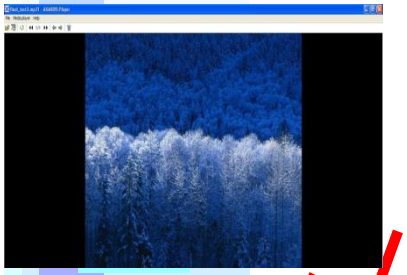
Scenario di test (1)

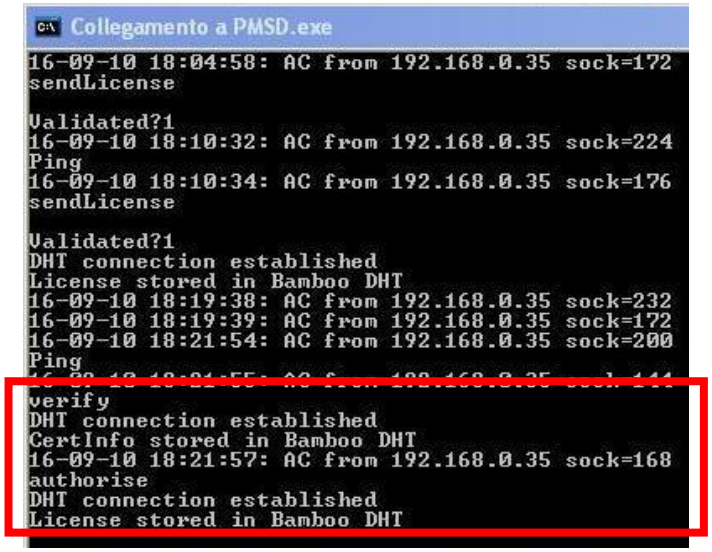


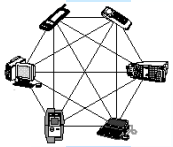


Analoga procedura usata per la certificazione e verifica del tool

Scenario di test (2)

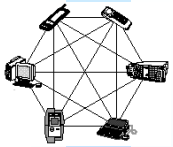






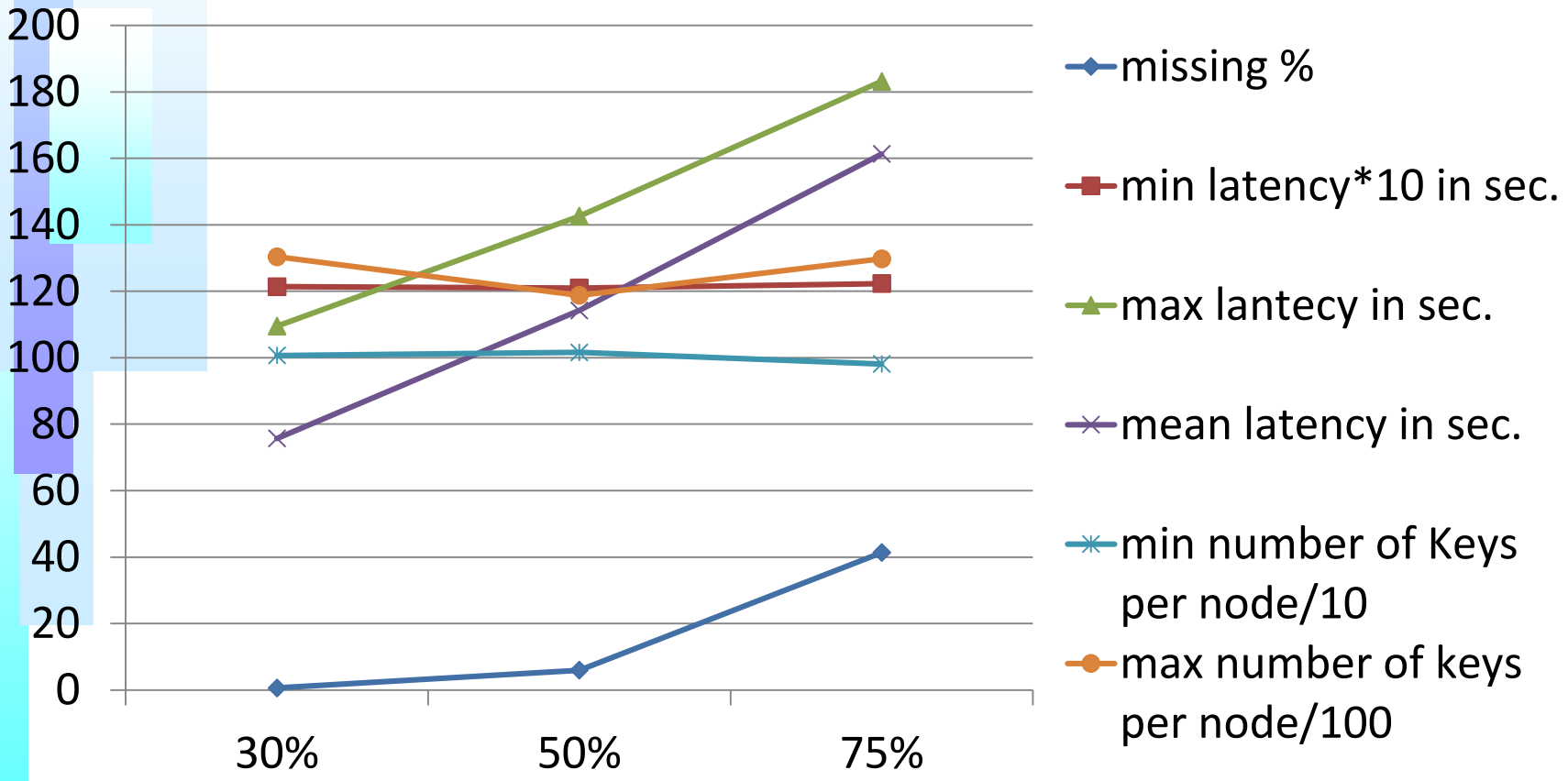
Considerazioni su P2P DRM

	DRM AXMEDIS TRADIZIONALE	DRM P2P
SICUREZZA	SI	SI
TRUST MANAGEMENT	SI	SI
RIDONDANZA DEI DATI	NO	SI
SCALABILITA'	NO	SI
EFFICIENZA	Lineare	Logaritmica
Costo di storage	O(L=C*U)	O(L=C*U) + duplications on DHT
Costo di accesso, transazione	O(U*devices), caso peggiore	O(1) → O(U*D) sul server oppure su DHT O(Log2 N)
FAULT TOLERANCE	NO	SI

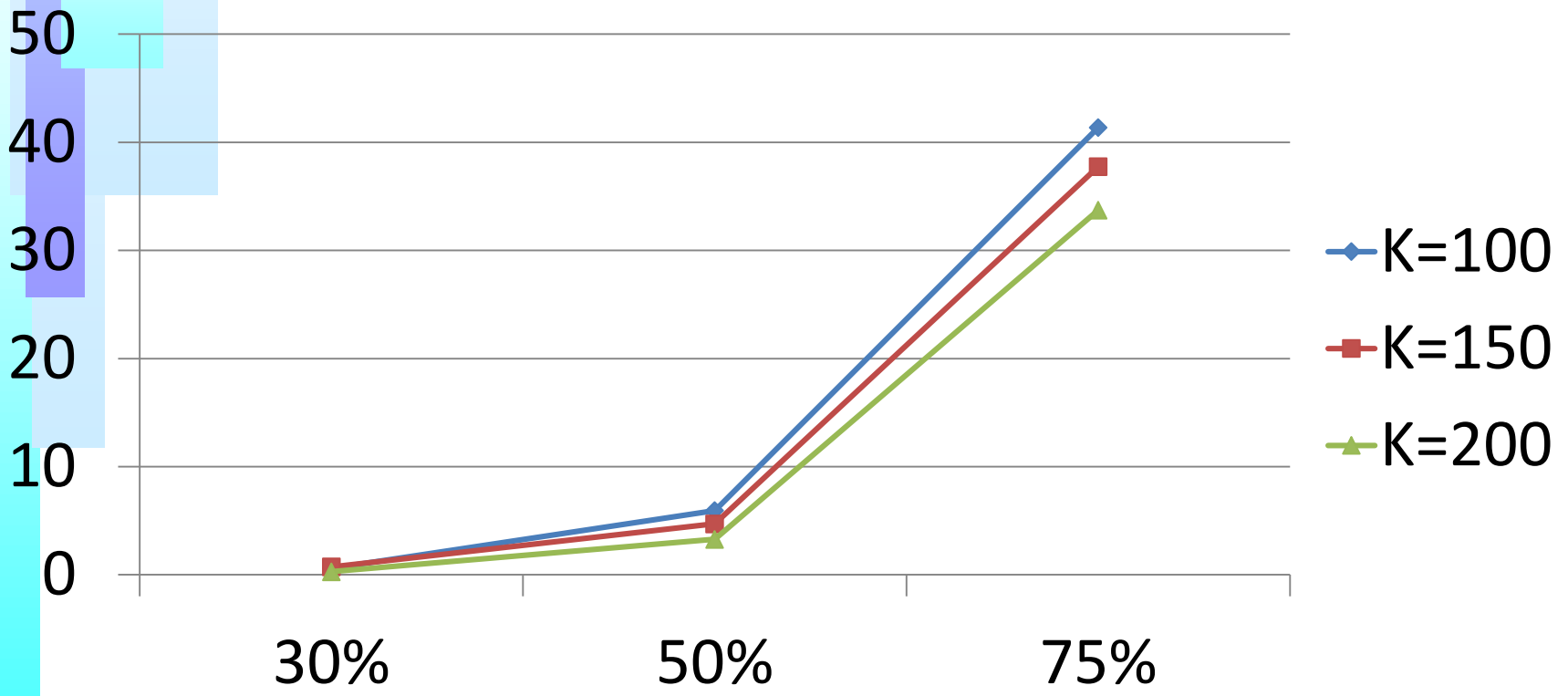


Parameter	Description	Range of values
L	Size of the leaf node list	8-400
Ltime	Time to update the leaf node list	25-250s
K	Number of replicas per node	4-200
N	Number of nodes involved in the network	1000-10000
Delay	Delay of transmission	30-50 ms
Jitter	variability over time of the packet latency across a network	10%
SendQueueLengh	Max dimension of the output buffer	0.5 Mbyte

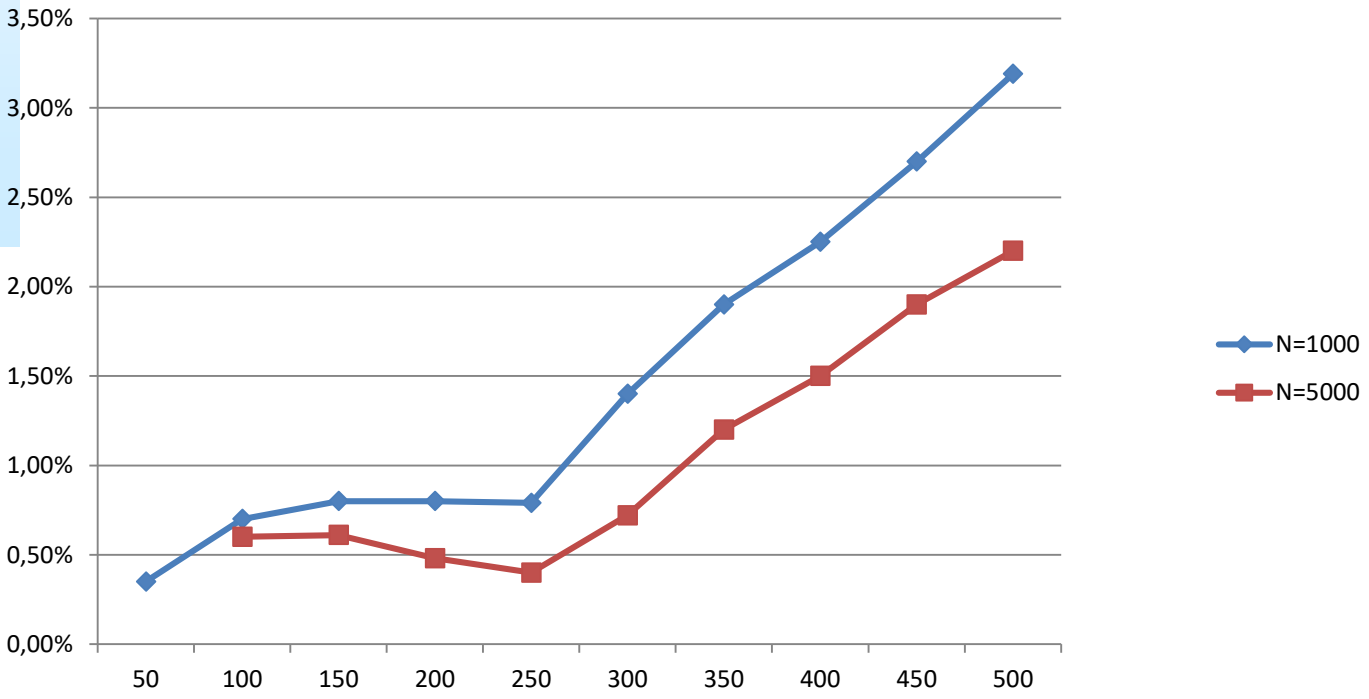
(a) L=200, K=100;


200
180
160
140
120
100
80
60
40
20
0
30%
50%
75%
missing %
min latency*10 in sec.
max lantecy in sec.
mean latency in sec.
min number of Keys per node/10
max number of keys per node/100

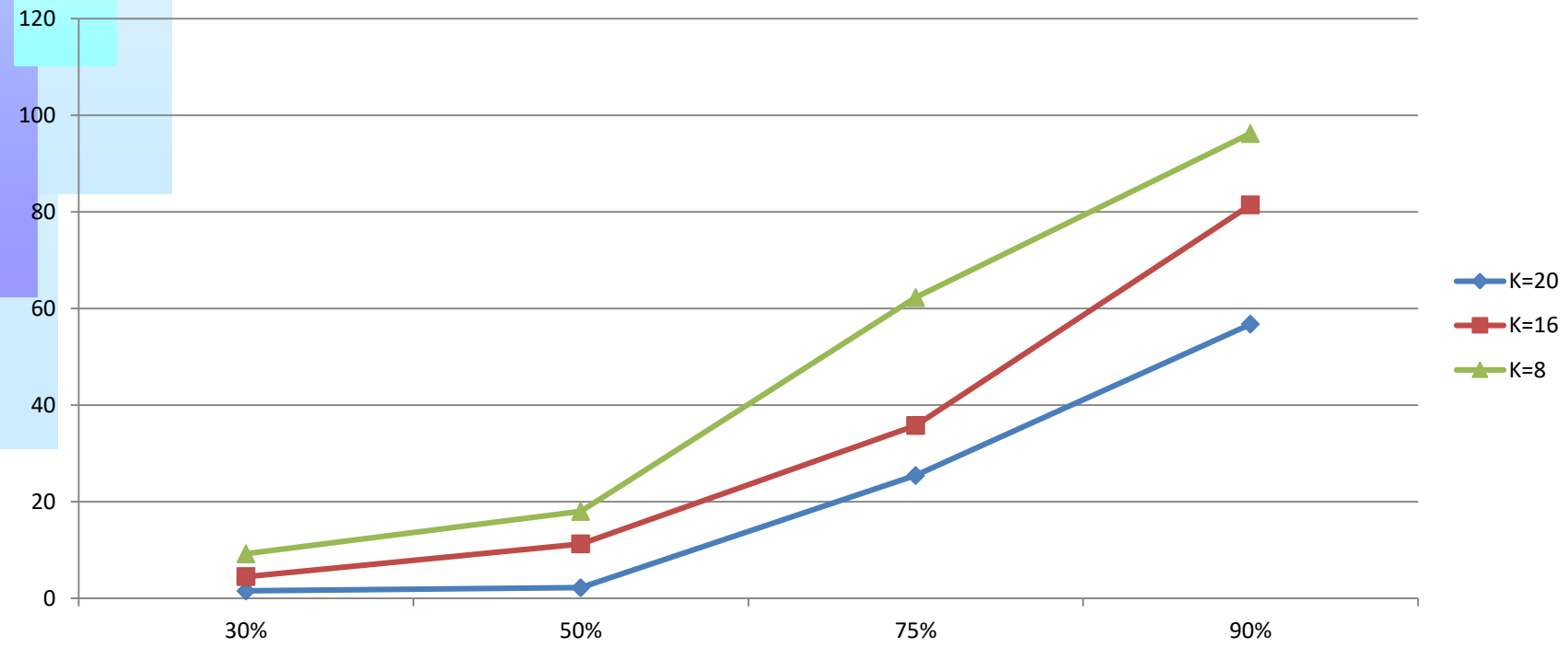
Trend of the percentage of missing estimated on the DHT P2P for the cases of a churn of 30%, 50% and 75%, and K equal to 100, 150, and 200.



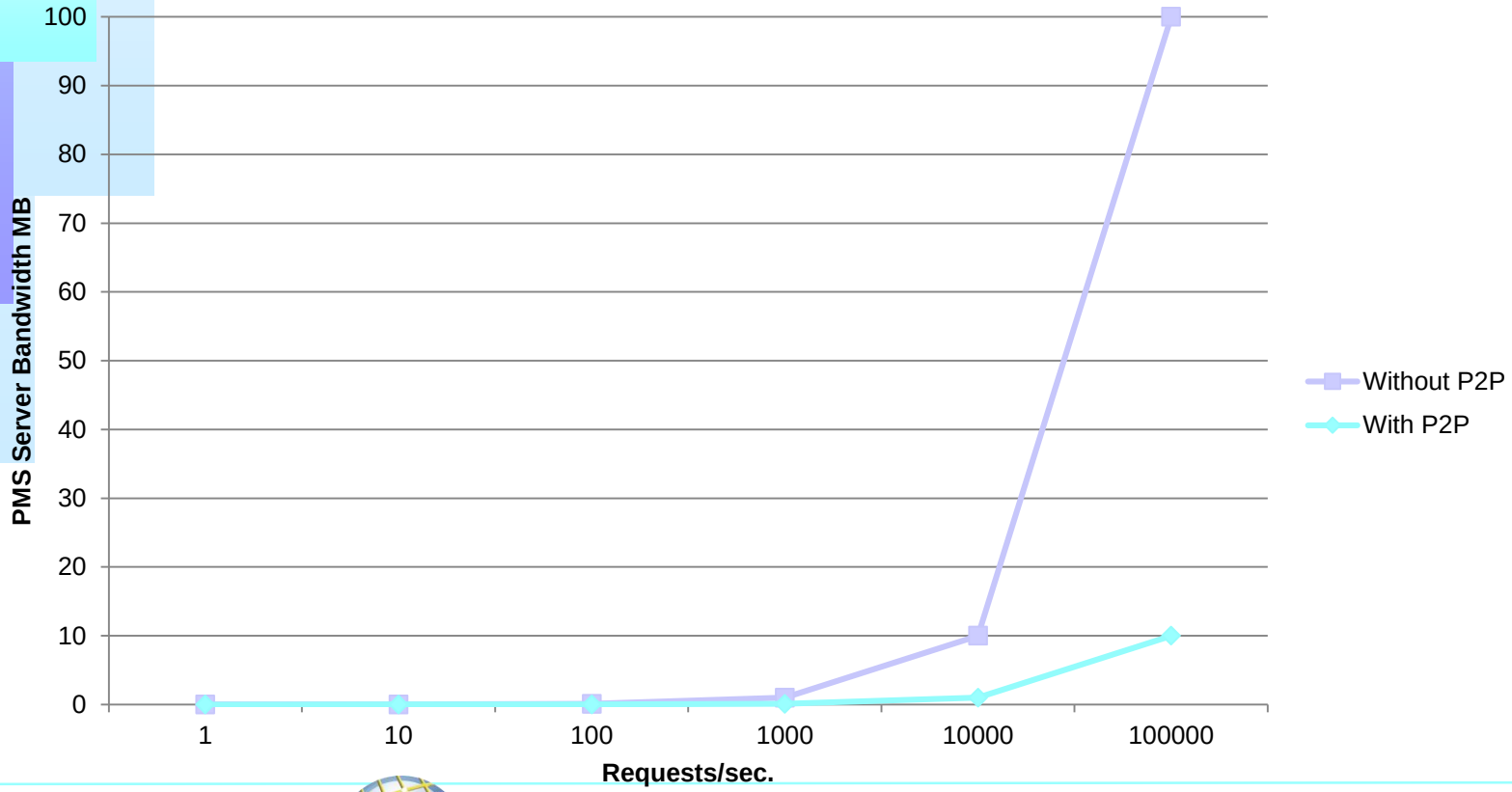
Trend of the percentage of missing estimated on the DHT P2P With respect to the Ltime value for the cases of a churn of 30%, and different values of the number of nodes, N, with the same number of replica

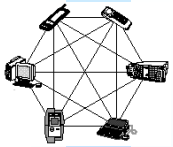


Trend of the percentage of missing estimated on the DHT P2P with respect to different percentage of churn, for *K* equal to 20, 16 and 8



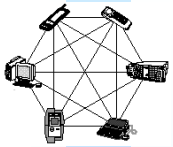
Trend of needed bandwidth for the PMS Server with respect to the number of requests per second of verification/authentication coming from the clients both with and without P2P solution, for K=100.





Bibliografia

- P. Bellini, P. Nesi, F. Pazzaglia, "Exploiting P2P Scalability for Grant Authorization in Digital Rights Management Solutions", International Journal Multimedia Tools and Applications, Springer press, 2013. , DOI 10.1007/s11042-013-1468-y, Pub on line April 2013,
- Emanuele Bellini , Paolo Nesi, "A Trust P2P network for the Access to Open Archive resources", WORLD LIBRARY AND INFORMATION CONGRESS: 75TH IFLA GENERAL CONFERENCE AND COUNCIL, 23-27 August 2009, Milan, Italy, http://www.ifla.org/annual-conference/ifla75/index.htm
- P. Bellini, I. Bruno, D. Cenni, P. Nesi, D. Rogai, “P2P Architecture for Automated B2B Cross Media Content Distribution”, Automated Production of Cross Media Content for Multi-Channel Distribution, 2007. AXMEDIS '07. Third International Conference on, AXMEDIS 2007, IEEE press, 28-30 Nov. 2007 Page(s):105 - 112, Digital Object Identifier 10.1109/AXMEDIS.2007.31



Bibliography

- Colouris Book come menzionato nelle prime slide del corso
- progetto Jxta (http://www.jxta.org)
- progetto MyJxta2 (http://myjxta2.jxta.org)
- Bernard Traversa, *Project JXTA 2.0 Super-Peer Virtual Network*, Maggio 2003.
- AXMEDIS P2P solution extending BitTorrent
  - www.axmedis.org