Peer-to-peer networks

How to perform offloading of download/upload/sharing/... applications

Learning outcomes

- Understand the concept(s) of P2P and its diversity
- Compare diferente types of models, from centralized to fully distributed
- Realize that P2P is a design option that may be performed at diferente levels (searching, identification, routing,…)
- Understand the overall concept of a DHT, and its operation
- Identify some P2P techniques.

70

Contents

- The P2P model
- P2P terminology
- P2P networks exemples
- Issues in P2P networks: routing, searching
- DHTs

71

Peer to peer networks

- Specific type of content distribution networks, without (or minimal) central control
- Peer is "…an entity with capabilities similar to other entities in the system."

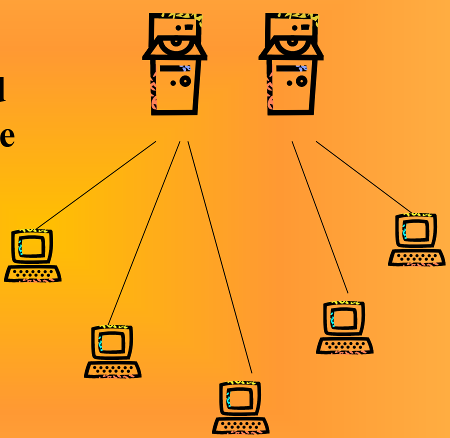
72

Peer to peer networks

- Exploits diverse connectivity between participants in a network
- Exploits the cummulative bandwidth of network participants
- Typically used for connecting nodes via largely ad-hoc connections
  - Sharing content files containing audio, video, data
  - Even real-time data, such as telephony traffic, is also passed using P2P technology
- Pure peer-to-peer network
  - There is no notion of clients or servers
  - Equal peer nodes that simultaneously function as both "clients" and "servers" to the other nodes on the network

The Web Model

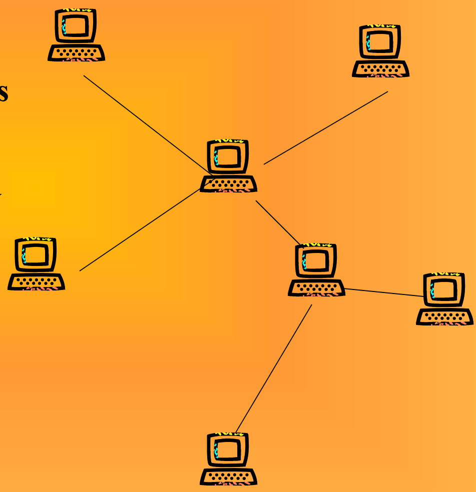
- Contact a server and download a web page
- Server has all the resources and capabilities



73

The P2P Model

- A peer's resources are similar to the resources of the other participants
- P2P – peers communicating directly with other peers and sharing resources
- P2P services
  - Distributed Computing
  - File Sharing
  - Collaboration
  - Platforms



74

Advantages

75

- Clients provide resources, including bandwidth, storage space, and computing power
- As nodes arrive and demand on the system increases, the total capacity of the system also increases
- Distributed nature also increases robustness in case of failures by replicating data over multiple peers
  - Enable peers to find the data without relying on a centralized index server

76

P2P applications

- File sharing
  - Using application layer protocols
    - DirectConnect (centralized), Gnutella (flooding), BitTorrent (hybrid)
- VoIP
  - Using application layer protocols
    - SIP
- Streaming media
- Instant messaging
- Distributed Computing
  - SETI@home
- Software publication and distribution
- Media publication and distribution
  - radio, video

78

Challenges for P2P

Even considering the basic file transfer case:

- Peer discovery and group management
- Data location, searching and placement
  - Search and routing
- Reliable and efficient file delivery
- Security/privacy/anonymity/trust

➢ Design Concerns
  - Per-node state
  - Bandwidth usage
  - Search time
  - Fault tolerance/resiliency

79

Contents

P2P types

- **Pure P2P** refers to an environment where all the participating nodes are peers
  - No central system controls, coordinates, or facilitates the exchanges among the peers
- **Hybrid P2P** refers to an environment where there are servers which enable peers to interact with each other
  - The degree of central system involvement varies with the application
  - Different peers may have different functions (simple nodes, routers, rendezvous)

No method is better than the other.

- each has its advantages and its drawbacks, each is the right choice for some applications

81

Aspects of P2P

- Types of Peers
- Centralized vs Distributed
- Structured versus Unstructured
- Levels of discussion
  - Discovery / Storage
  - Information location
  - Location of information

82

Types of Peers

- There are three types of peers, in general:
  - Simple peers
  - Rendezvous peers
  - Router/Relay peers

Simple Peers

- A simple peer is designed to serve a single end user, allowing that user to provide services from his device and consuming services provided by other peers on the network.
  - Normally,
    - a simple peer will be located behind a firewall, separated from the network at large;
    - peers outside the firewall will probably not be capable of directly communicating with the simple peer located inside the firewall.
  - Because of their limited network accessibility, simple peers have the least amount of responsibility in any P2P network.
- Simple peers are not responsible for handling communication on behalf of other peers or serving third-party information for consumption by other peers.

Rendezvous Peers

Taken literally, a rendezvous is a gathering or meeting place;

- in P2P, a rendezvous peer provides peers with a network location to use to discover other peers and peer resources.
- usually outside a private internal network's firewall.
- A rendezvous could exist behind the firewall, but it would need to be capable of traversing the firewall using either a protocol authorized by the firewall or a router peer outside the firewall.

- Peers issue discovery queries to a rendezvous peer, and the rendezvous provides information on the peers it is aware of on the network.
- A rendezvous peer can augment its capabilities by caching information on peers for future use or by forwarding discovery requests to other rendezvous peers.
  - These schemes have the potential to improve responsiveness, reduce network traffic, and provide better service to simple peers.

Router (Relay) Peers

- A router peer provides a mechanism for peers to communicate with other peers separated from the network by firewall or Network Address Translation (NAT) equipment.
- A router peer provides a go-between that peers outside the firewall can use to communicate with a peer behind the firewall, and vice versa.
- To send a message to a peer via a router, the peer sending the message must first determine which router peer to use to communicate with the destination peer.
  - A relay is not necessarily a rendezvouz peer: the former is on the data stream, while the later is always on the discovery path (and maybe in the data stream).

Structured vs Unstructured

86

- **Based on how the nodes in the overlay network are linked to each other**
  - **Unstructured P2P networks**
    - Formed when the overlay links are established arbitrarily.
    - If a peer wants to find a desired piece of data in the network, the query has to be flooded through the network to find as many peers as possible that share the data
      - The queries may not always be resolved
        - » If a peer is looking for rare data shared by only a few other peers, then it is highly unlikely that search will be successful
      - Flooding causes a high amount of signalling traffic in the network
      - Gnutella and FastTrack/KaZaa, BitTorrent
  - **Structured P2P networks**
    - Employs a globally consistent protocol (logic) to ensure that any node can efficiently route a search to some peer that has the desired file, even if the file is extremely rare
    - The most common type of structured P2P network is the Distributed Hash Table (DHT)
      - A variant of consistent hashing is used to assign ownership of each file to a particular peer, in a way analogous to a traditional hash table's assignment of each key to a particular array slot
      - Chord, Pastry, Tapestry, CAN, Tulip, Kademlia

87

Centralized vs Distributed

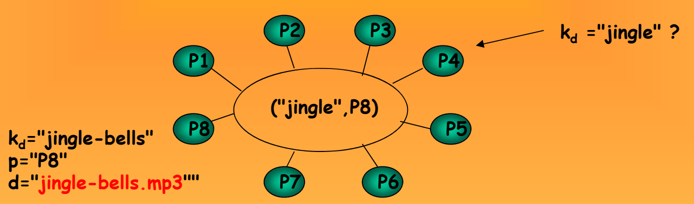
- **Centralized**
  - There is a centralized server for the upload and download of files or other applications
- **Flooding/distributed**
  - Any node in the p2p network can work as a peer server for download/upload
- **Hybrid**
  - Both centralized and distributed approaches, for redundancy
  - It is chosen the closer client
  - Load balancing is used

Note: discussion may be about the searching algorithm as well!

Resource Location in P2P Systems

- **Problem: Peers need to locate distributed information**
  - Peers with address p store data items d that are identified by a key k_d
  - Given a key k_d (or a predicate on k_d) locate a peer that stores d, i.e. locate the *index information* (k_d, p)
  - Thus, the data we have to manage consists of the key-value pairs (k_d, p)

- **Can such a distributed database be maintained and accessed by a set of peers without central control ?**

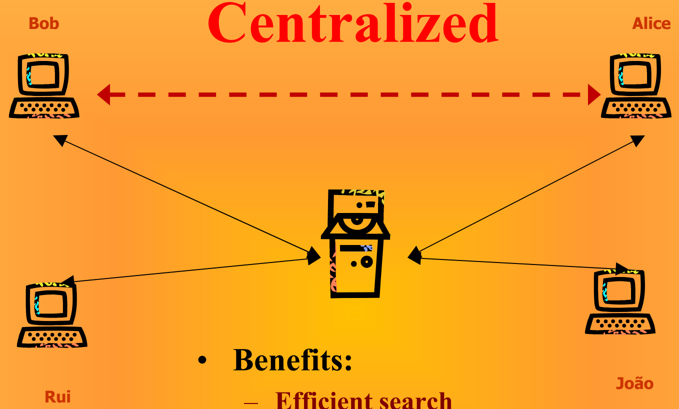


89

P2P Discovery Algorithms

- **Centralized Directory Model (CDM)**
  - The peers connect to a central directory where they publish informations about the shared content
  - Upon request from a peer, the central index will find the best peer that matches the request
  - Advantages: simple, high degree of control on shared contents
  - Limits: not scalable, single point of failure.
    - E.g.: Napster, Direct Connect, eDonkey (eMule). BitTorrent
- **Flooded Requests Model (FRM)**
  - Pure P2P algorithm in which each request from a peer is flooded (broadcasted) to directly connected peers, which themselves flood their peers, etc.
  - Advantages: efficient in limited communities (i.e. not very scalable).
  - Limits: requires large bandwidth.
    - E.g.: Gnutella, FastTrack
- **Document Routing Model (DRM)**
  - This algorithm is based on Distributed Hash Tables (DHT).
  - Publishing of a document: routing it to the peer whose ID is the most similar to the document ID, and repeating the process until the nearest peer ID is the current peer's ID.
  - Discovery: the request goes to the peer whose ID is the most similar to the document ID, and the process is repeated until the document is found.
  - Advantages: scalable.
  - Limits: malicious participants can threaten the liveness of the system.
    - E.g.: Chord, Kademlia, FreeNet, CAN, Tapestry.

Centralized

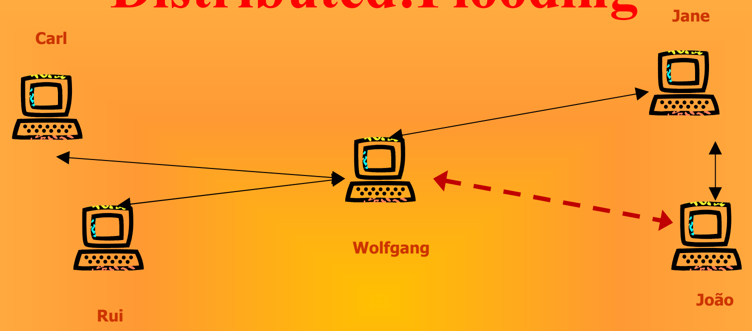


- **Benefits:**
  - Efficient search
  - Limited bandwidth usage
  - No per-node state
- **Drawbacks:**
  - Central point of failure
  - Limited scale
  - Infrastructure and administration costs

search
data transfer

93

Distributed:Flooding



- **Benefits:**
  - No central point of failure
  - Limited per-node state
  - No administration costs
- **Drawbacks:**
  - Slow searches
  - Bandwidth intensive

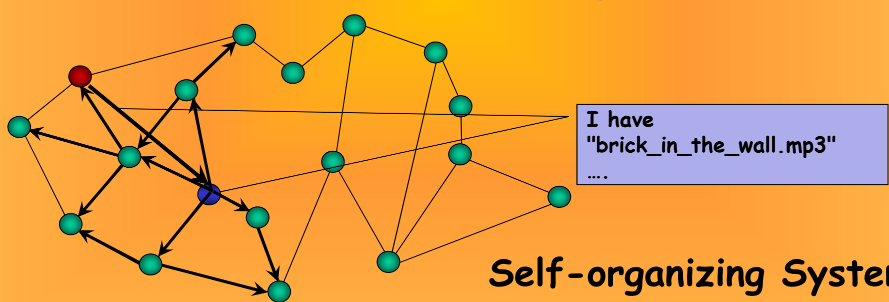
search
data transfer

94

Fully Decentralized Information Systems

- P2P file sharing
  - Global scale application
- Example: Gnutella
  - 40.000 nodes, 3 Mio files (August 2000)

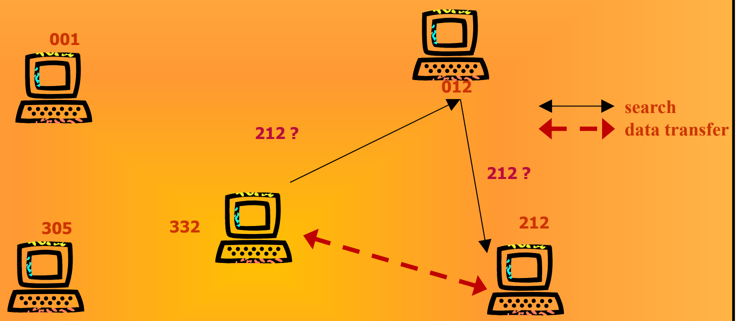
- Strengths
  - Good response time, scalable
  - No infrastructure, no administration
  - No single point of failure
- Weaknesses
  - High network traffic
  - No structured search
  - Free-riding



Self-organizing System

Gnutella: no servers

Hybrid/Hierarchical



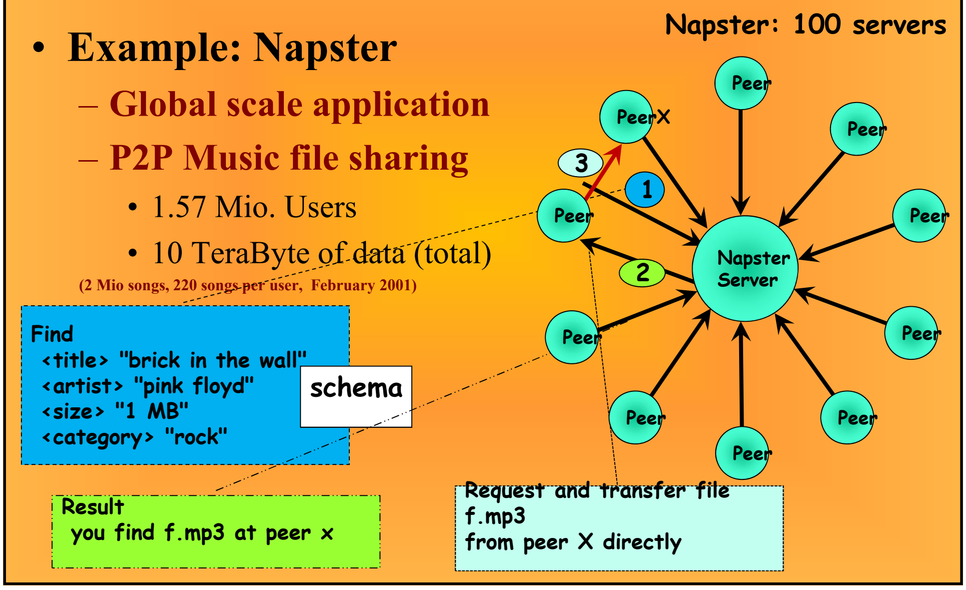
- **Benefits:**
  - No central point of failure
  - Limited per-node state
  - More efficient searching
  - Load sharing
- **Drawbacks:**
  - More complex nodes

96

(Semi-)Decentralized Information Systems

- Example: Napster
  - Global scale application
  - P2P Music file sharing
    - 1.57 Mio. Users
    - 10 TeraByte of data (total)

(2 Mio songs, 220 songs per user, February 2001)



Lessons Learned from Napster

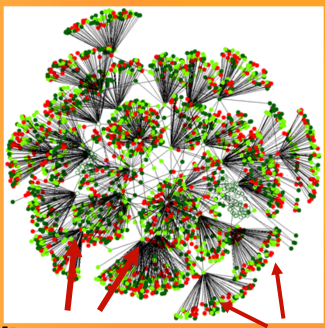
- **Strengths: Resource Sharing**
  - Every node "pays" its participation by providing access to its resources
    - physical resources (disk, network), knowledge (annotations), ownership (files)
  - Every participating node acts as both a client and a server ("servent"): P2P
  - global information system without huge investment
  - decentralization of cost and administration = avoiding resource bottlenecks
- **Weaknesses: Centralization**
  - server is single point of failure
  - unique entity required for controlling the system = design bottleneck
  - copying copyrighted material made Napster target of legal attack

Contents

- The P2P model
- P2P terminology
- **P2P networks exemples**
- Issues in P2P networks: routing, searching
- DHTs

Real Gnutella Network

Oct 2003 Crawl of public gnutella (v.2)



Ultrapeer nodes

Leaf nodes

>100 Files
0 Files
0-100 Files

- Popular open-source file-sharing network
  - ~450,000 users as of 2003
  - ~2,000,000 today
- Ultrapeer-based Topology
  - Queries flooded among ultrapeers
  - Leaf nodes shielded from query traffic
  - Based on multiple crawlers

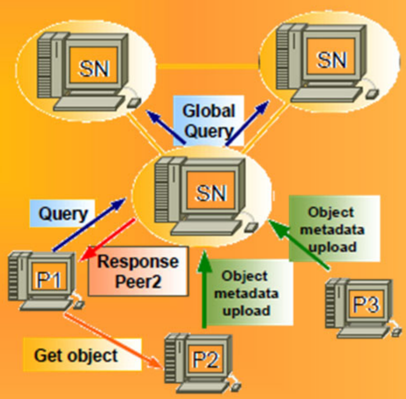
Direct Connect

- Composed by
  - Hubs,
  - Clients
  - HubListServer
- Hubs are naming services and communication facilitators for Clients
- HubListServer act as a naming services: clients discover Hubs asking the List Server
- Network architecture: Hybrid unstructured
- Algorithm: Centralized Directory Model (CDM)

104

FastTrack/KaZaA

- Extension of the Gnutella protocol
- Adds super-nodes to improve scalability (~gnutella v.2)
  - A peer application hosted by a powerful machine with a fast network connection become automatically a super-node, effectively acting as a temporary indexing server for other slower peers
  - Communicate between each others in order to satisfy search requests
- Network architecture: Hybrid Unstructured.
- Algorithm: Flooded Requests Model (FRM)

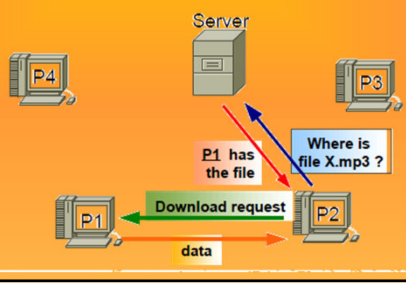


105

© Rui L. Aguiar (ruilaa@det.ua.pt) - Uni. Aveiro

Napster/OpenNAP

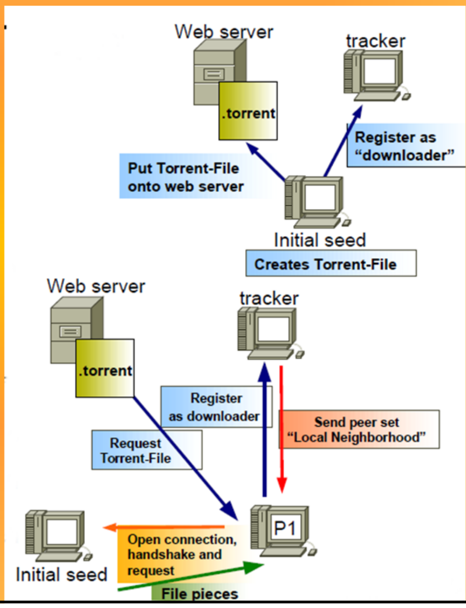
- Files (music) are on the cliente machine
- Servers provide search (rendezvous) and initiate direct transfers between clientes
- OpenNAP is an extension to other types and linking servers.
- Network architecture: Hybrid Unstructured.
- Algorithm: Centralized Directory Model (CDM)



106

BitTorrent

- Trackers are responsible for helping downloaders find each other, using a simple protocol on top of HTTP.
- A downloader sends status info to trackers, which reply with lists of contact information for peers which are downloading the same file.
- BitTorrent cuts files into pieces, which are broken into sub-pieces (chunks).
- The (Web) servers don't have information about content location
  - Only store metadata files describing the objects (length, name, etc.) and associating to each of them the URL of a tracker
- Network architecture: Hybrid unstructured
- Algorithm: Centralized Directory Model (CDM)



107

Contents

- The P2P model
- P2P terminology
- P2P networks exemples
- **Issues in P2P networks: routing, searching**
- DHTs

108

Issues in P2P

© Rui L. Aguiar (ruilaa@det.ua.pt) - Uni. Aveiro

- The discussions may reflect aspects of:
  - Information index storage
    - How and where to store the information indexes (the data that identifies where files are stored)
  - Information index location
    - How to retrieve the information índex, and locate the computer that contains the file(s)
  - Information storage
    - How and where to store the information (the files)
  - Information retrieval
    - How to retrieve the file, how to route request to the node

Information index is a file per itself

109

Routing

- IP routing in P2P networks is either centralised or (administratively) automatically configured and is therefore unproblematic
- P2P overlay network routes separately
  - Virtual topology on top of the physical links of the network
  - Nodes leave and join this network dynamically and the average uptime of individual nodes is relatively low
  - Topology may change all the time
  - Routing in these networks is therefore very problematic
- Some issues on designing P2P routing algorithms
  - Scalability
  - Complexity
  - Anonymity

Routing in Gnutella (structureless)

110

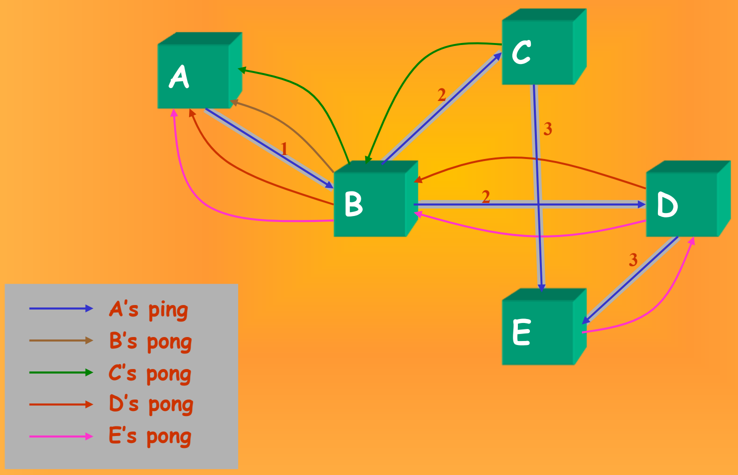
- The first mainstream overlay network
- Works well for small to medium sized networks

- To join the network a client has to know the address of at least one node already on the network
- Once the client has a connection to this node, it can then broadcast a ping to find the addresses of other nodes
- Each node maintains a connection to a number of other nodes, normally about five
- To search the network for a resource, the client sends a "Query" message to each of the nodes it is connected to
- They then forward the message
  - When a resource is found, the result i.e. resource name and address, is propagated back along the path
  - The number of nodes that get queried can be controlled using a Time-To-Live counter
  - Simplest kind of routing possible for an overlay network
- Searching in a Gnutella network is roughly of exponential complexity, as each search will take about n^d steps where n is the time to live and d is the number of peers per node

Gnutella: Protocol Message Types

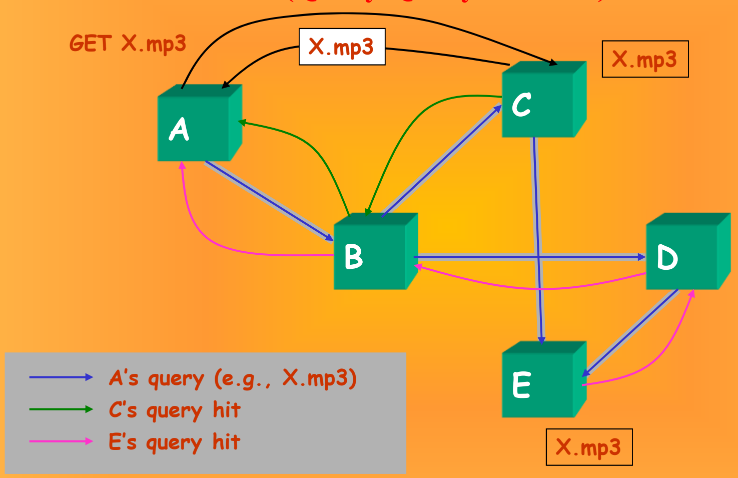
Type	Description	Contained Information
Ping	Announce availability and probe for other servents	None
Pong	Response to a ping	IP address and port# of responding servent; number and total kb of files shared
Query	Search request	Minimum network bandwidth of responding servent; search criteria
QueryHit	Returned by servents that have the requested file	IP address, port# and network bandwidth of responding servent; number of results and result set
Push	File download requests for servents behind a firewall	Servent identifier; index of requested file; IP address and port to send file to

Gnutella: Meeting Peers (Ping/Pong)



Gnutella: Searching

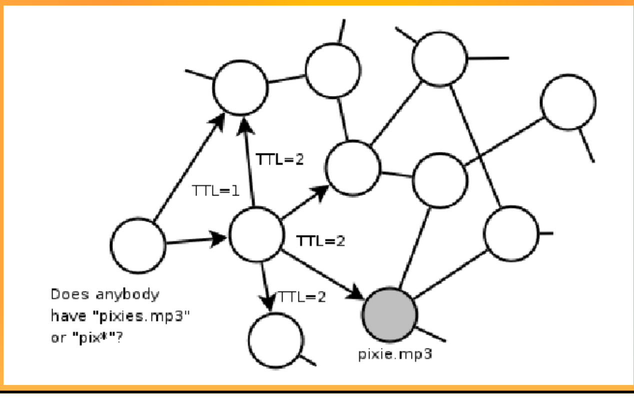
(Query/QueryHit/GET)



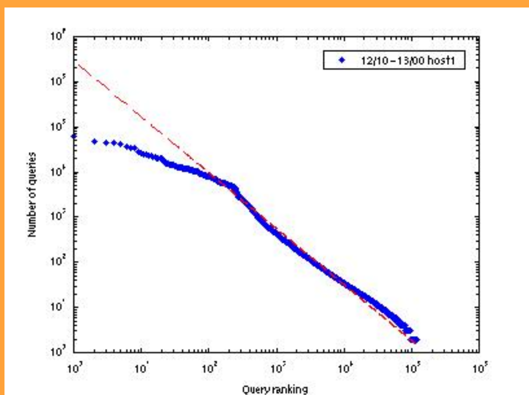
Searching in Gnutella (structureless)

114

- Queries are flooded to neighbours, have a TTL, and are forwarded only once
- Query may obtain several responses indicating which peers provides the requested file. Among those it selects one, and directly contacts it in order to download the file.
  - Can we search using fewer packets?



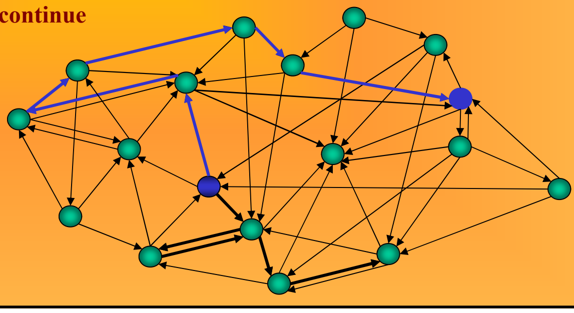
Popularity of Queries [Sripanidkulchai01]



- Very popular documents are approximately equally popular
- Less popular documents follow a Zipf-like distribution (i.e., the probability of seeing a query for the i^{th} most popular query is proportional to $1/(i^{alpha})$)
- Access frequency of web documents also follows Zipf-like distributions ⇒ caching might work for Gnutella

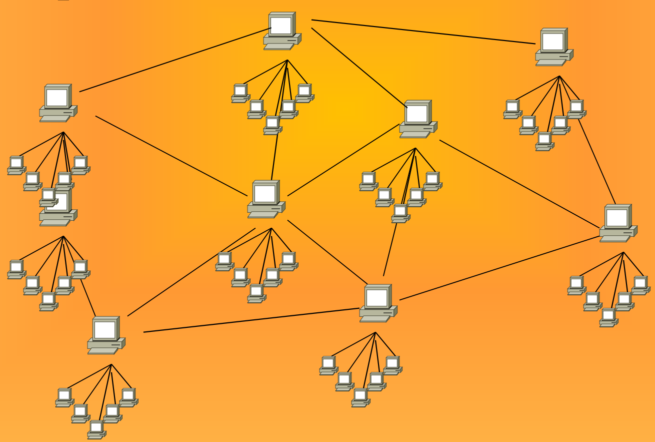
Improvements of Message Flooding

- **Expanding Ring**
  - start search with small TTL (e.g. TTL = 1)
  - if no success iteratively increase TTL (e.g. TTL = TTL +2)
- **k-Random Walkers**
  - forward query to one randomly chosen neighbor only, with large TTL
  - start k random walkers
  - random walker periodically checks with requester whether to continue



Hybrid Gnutella: “Ultrapeers”

- **Ultrapeers can be installed (KaZaA) or self-promoted (Gnutella v.2)**



118

Freenet routing

Files on the system are referenced and located using hash-keys

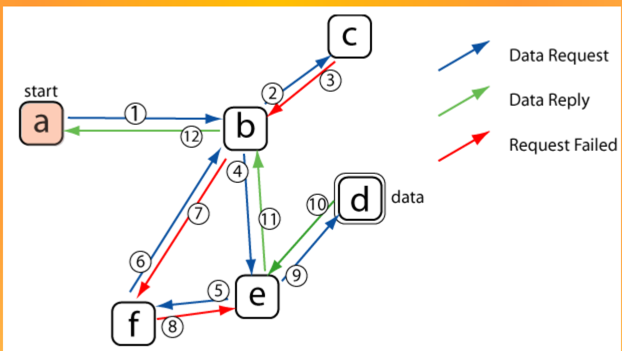
In order to search for a file, a user must know it's hash-key, or at least how to calculate it

- Once this is known, the user sends a request to their own freenet node
  - Each request has a hops-to-live value, and a randomly generated id so it can be recognised and rejected by nodes which have seen it before
- When a node receives a request, it first checks it's own data-store to see if it contains the relevant file
  - If it does, then it returns the file along with a message identifying it as the source of the data
  - If not, it decrements the hops-to-live value and forwards the request to one of the neighbours in its routing table
- If this request fails, then it asks another neighbour from the routing table
- If none of its neighbours return a positive response, then it returns a failed message itself
  - If the hops to live value is exceeded, then a failed message is also returned

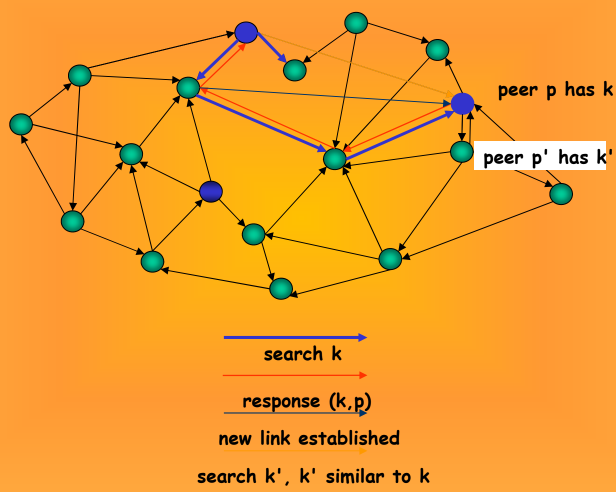
119

Freenet routing

- If a positive response is received from a node, then this node sends the data to the node it received the request from, and caches a copy of the data in its own data-store
- If the data-store is already full, then the least recently used files are deleted to make room for the new one



FreeNet Routing



- As responses are cached along each node, when the answer is returned, new links between the node holding the result and the nodes along the request paths are established.
- creates new, shorter paths, for future requests, which are similar (or the same).

Semantic routing

121

- Improves on traditional routing by prioritising nodes which have been previously good at providing information about the types of content referred to by the query
  - Nodes must have a constant identifier within the network's namespace if they are to retain their confidence ratings
- Each time a node answers a query, its peers adjust their confidence in that node with regard to that type of query
  - The nature of this adjustment depends upon whether the node answered correctly i.e. if the search result was selected by the searcher
  - The information associated with 'types' depends greatly on the kind of semantic data being dealt with by the network and the strictness of the peer confidence ranking algorithm
- There is an initial forming stage where none of the peers have ratings for any nodes, and nodes might be returned randomly (or indeed using traditional routing methods)

122

Early bird easter egg

- Qual a diferença entre redes P2P estruturadas e não estruturadas?
  - *What is the difference between structured and non-structured P2P networks?*

123

Contents

- The P2P model
- P2P terminology
- P2P networks exemples
- Issues in P2P networks: routing, searching
- **DHTs**

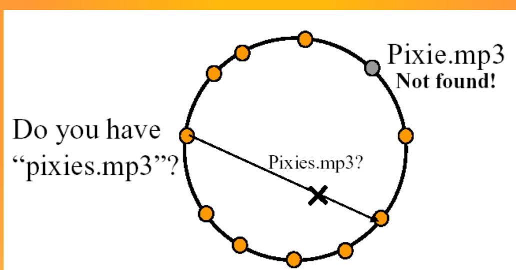
Resource Location Problem

- Operations
  - search for a key at a peer: p->search(k)
  - update a key at a peer: p->update(k,p')
  - peers joining and leaving the network: p->join(p')
- Performance Criteria (for search)
  - search latency: e.g. searchtime(query) ≈ Log(size(database))
  - message bandwidth, e.g. messages(query) ≈ Log(size(database))
    messages(update) ≈ Log(size(database))
  - storage space used, e.g. storagespace(peer) ≈ Log(size(database))
  - resilience to failures (network, peers)
- Qualitative Criteria
  - complex search predicates: equality, prefix, containment, similarity search
  - use of global knowledge
  - peer autonomy
  - peer anonymity and trust
  - security (e.g. denial of service attacks)

125

Searching in DHTs (structured)

- Need to know the exact filename
  - Keys (filenames) map to node-ids
    - Change in file name → search at different nodes
    - No wildcard matching: cannot ask for file "pix*"



126

Routing through DHT

- Useful for sharing files or other data across a peer-to-peer network
- A hash function takes a variable length string of bytes and returns a number that it generates from this
  - DHT algorithms work by hashing all file/data identifiers and storing their locations in a giant hash table, which is distributed across the participating nodes
- Chord is a good example of a DHT algorithm
  - All files/data items in the network will have an identifier, which will be hashed using this function to give a key for that particular resource
  - If a node needs a file/data, they will hash its name and then send a request using this key.
  - All *n* nodes also use this function to hash their IP address, and conceptually, the nodes will form a ring in ascending order of their hashed IP
  - When a node wants to share a file or some data
    - Hashes the identifier to generate a key, and sends its IP and the file identifier to successor(key)
    - These are then stored at this node
    - All resources are indexed in a large DHT across all participating nodes
    - If there are two or more nodes that hold a given file or resource, the keys will be stored at the same node in the DHT, giving the requesting node a choice.

127

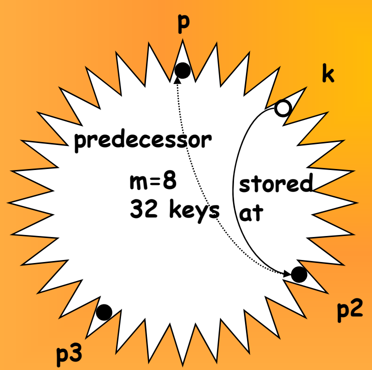
Routing through DHT

- When a node wants something
  - Hashes its identifier and sends a request to successor(key)
  - Reply with the IP of the node that holds the actual data
  - How does a node request information from successor(key), when it doesn't know its IP, but only the key?
    - Every node holds what is known as a finger table
      - Contains a list of keys and their successor IP's, and is organized such that each node holds the IP of an exponential sequence of nodes that follow it, i.e. entry *i* of node *k*'s finger table holds the IP of node $k + 2^i$
  - Searching for a node is a log(n) procedure
    - Each node knows the IP of the next real node in the ring
    - If the key lies between k and the next real node, then successor(key) is the next node
    - Otherwise, the finger table is searched for the closest predecessor of the key
    - The request is forwarded to this node and it repeats the same procedure until the node is found
    - The request contains the IP of the requesting node, the reply can be sent instantly, with no back propagation required
- Advantages
  - Guarantee that a reply will arrive within log(n) time
  - Lack of redundant overhead

Distributed Hash Tables

- Hashing of search keys AND peer addresses on binary keys of length m
  - e.g. m=8, key("jingle-bells.mp3")=17, key(196.178.0.1)=3

Data keys are stored at next larger node key



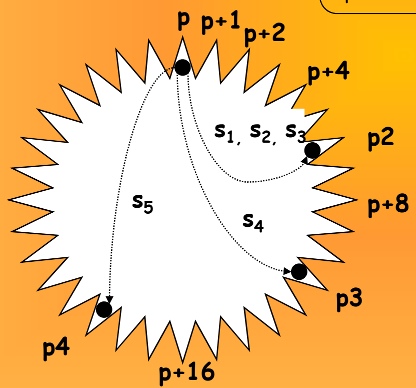
peer with hashed identifier p,
data with hashed identifier k, then
k ∈] predecessor(p), p]

Search possibilities
1. every peer knows every other
   O(n) routing table size
2. peers know successor
   O(n) search cost

Routing Tables

Every peer knows m peers with exponentially increasing distance

Each peer p stores a routing table
First peer with hashed identifier s_i such that
s_i =successor(p+2^{i-1}) for i=1,…,m

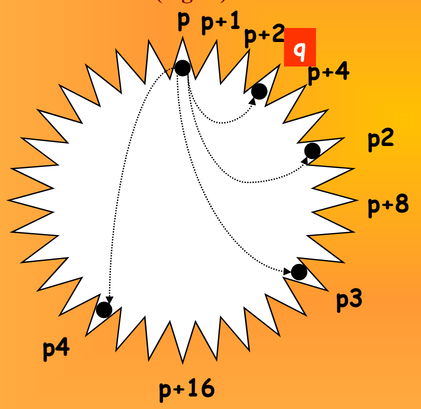


i	s_i
1	p2
2	p2
3	
4	p3
5	p4

Search
O(log n) routing table size

Node Insertion

- New node q joining the network
  - q asks existing node p to find predecessor and fingers
  - cost: $O(\log^2 n)$



routing table of p

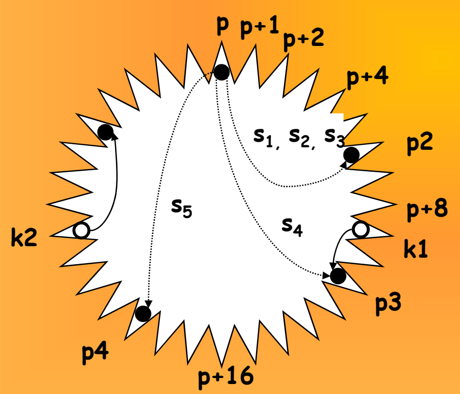
i	s_i
1	q
2	q
3	p2
4	p3
5	p4

routing table of q

i	s_i
1	p2
2	p2
3	
4	p3
5	p4

Search

```
search(p, k)
find in routing table largest (i, p*) such that p* ∈ [p,k[
/* largest peer key smaller than the searched data key */
if such a p* exists then search(p*, k)
else return (successor(p)) // found
```



Search
  $O(\log n)$ search cost

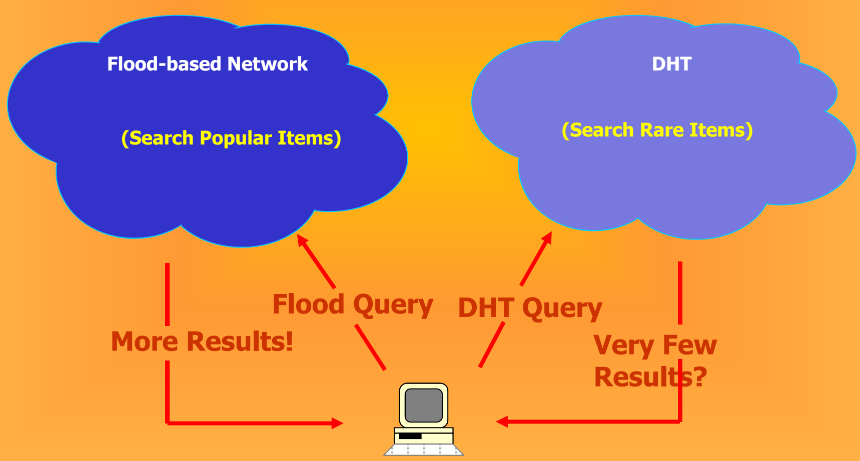
Routing Table with exp. increasing distance ⇒ $O(\log n)$ with high probability

File Search: Flooding vs. DHTs

- Recall
  - Flooding can miss files
  - DHTs should never
- Query complexity
  - Flooding can handle arbitrary single-site logic
  - DHTs can do equijoins, selections, aggregates, etc.
    - But not so good at fancy selections like wildcards
- Query Performance
  - Flooding can be slow to find things, uses lots of BW
  - DHTs: expensive to publish documents with lots of terms
  - DHTs: expensive to intersect really long term lists
    - Even if output is really small!
    - Not likely to replace Google any time soon
- Hybrid solution!

Hybrid Search

Hybrid = “Best of both worlds”



134

FYI - Security aspects: attacks

- Poisoning attacks
  - e.g. providing files whose contents are different from the description
- Polluting attacks
  - e.g. inserting "bad" chunks/packets into an otherwise valid file on the network
- Freeloaders
  - Users or software that make use of the network without contributing resources to it
- Insertion of viruses to carried data
  - e.g. downloaded or carried files may be infected with viruses or other malware
- Malware in the peer-to-peer network software itself
  - e.g. distributed software may contain spyware
- Denial of service attacks
  - Attacks that may make the network run very slowly or break completely
- Filtering
  - Network operators may attempt to prevent peer-to-peer network data from being carried
- Identity attacks
  - e.g. tracking down the users of the network and harassing or legally attacking them
- Spamming
  - e.g. sending unsolicited information across the network- not necessarily as a denial of service attack

135

FYI: Security measures

- Most attacks can be defeated or controlled by careful design of the peer-to-peer network and through the use of encryption
  - However, almost any network will fail when the majority of the peers are trying to damage it
- Anonymity
  - Some peer-to-peer protocols (such as Freenet) attempt to hide the identity of network users by passing all traffic through intermediate nodes
- Encryption
  - Some peer-to-peer networks encrypt the traffic flows between peers
    - Make it harder for an ISP to detect that peer-to-peer technology is being used (as some artificially limit bandwidth)
    - Hide the contents of the file from eavesdroppers
    - Impede efforts towards law enforcement or censorship of certain kinds of material
    - Authenticate users and prevent 'man in the middle' attacks on protocols
    - Aid in maintaining anonymity

To Conclude:

P2P and self-organization in Architectures

P2P self-organization concepts can be found at every layer, reflecting some sort of self-organization in the communication structure.

Layer	Communication type	Information discovery	Information transport
(at which Layer we are considering self-organization)	*Communication concept explored*	*How to figure where the information is*	*How is the information encapsulated for transport*
Networking Layer	Regular Internet (IP) protocols	Routing, DNS	TCP
Data Access Layer	Overlay Networks, P2P	Resource Location (DHT, central server)	Gnutella, FreeNet
Service Layer	Application interface	Messaging, Distributed Processing	Napster, SETI, Groove
User Layer	User Communities, Google Circles	Collaboration	eBay, Google+, Facebook,