Introduction on Peer to Peer systems

Georges Da Costa
dacosta@irit.fr

Routing

First generation system

Queries are sent to all the places where the data could be

- Gnutella : Flooding the whole distributed index
- Napster : Directly to the whole centralized index

Next generation system

Queries are sent to a node which might know a way to the data

A case study : Freenet

Ian Clarke, University of Edinbourgh, (1999)

Keywords

- A peer-to-peer file sharing system
- Provide anonymity for authors and readers
- A web of Freedom

- Principle
  - Files are referenced by key
  - The key is obtained by SHA-1 on the file
  - The key is routed to localize the file

Content Driven routing algorithm

- Routing table contains a set of key/node pairs
- Take the nearest key in the routing table to obtain the next node to consult.
- Nearest key = by lexical comparison

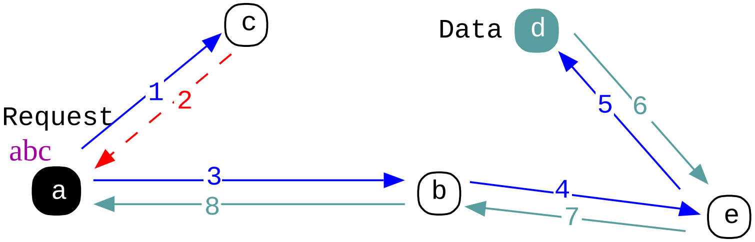


abb	node c
acd	node b

On the path of the answer

- File is replicated on the path in the cache
- Cache : variant of *L*ast *R*ecently *U*sed
- Routing tables are updated

$\rightarrow$ the graph evolves (new links = new entries)

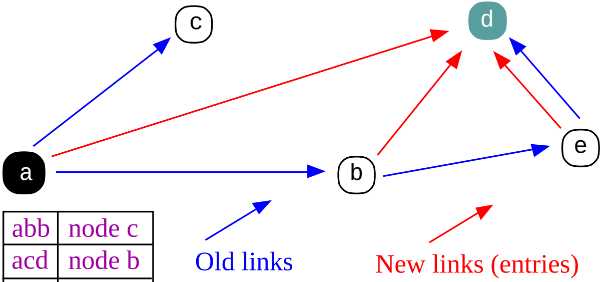


abb	node c
acd	node b
abc	node d

Anonymity

- Reader
  - Impossible to know if a user is forwarding or initiating the request
  - Impossible to know if a user is the last to receive a file
- Writer
  - Once in the system, the writer can disconnect
  - Impossible to know if someone insert some file or forward it

Some properties

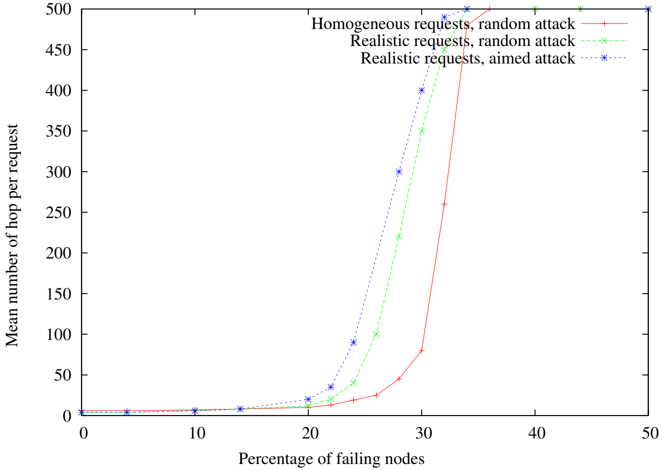
Self-organization of the graph

- Nodes specialize in files with close keys (learning process)
- Good properties (Small World)

File are automatically replicated in function of their popularity

- Hot-spots are limited
- Tolerant against attacks

Fault Tolerance



Drawbacks

Counterpart

- Files might disappear (LRU cache)
- The network is heavily loaded
- Difficult to update a value
- Impossible to know what is hosted locally

Pastry

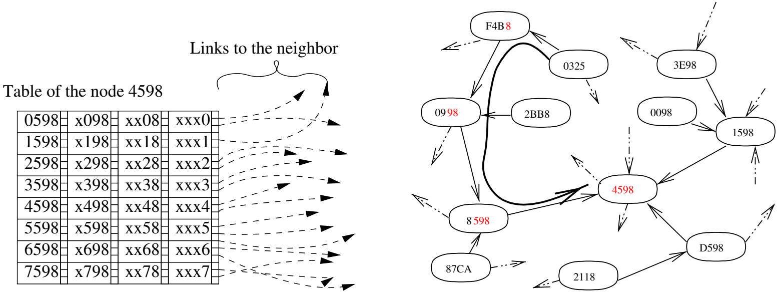
Principle

- Each file has a key
- Each node has an identifier
- Node with identifier *Id* manages keys whose values are near *Id*

Queries

- Content driven queries
- Suffix forwarding

Pastry II



0598	x098	xx08	xxx0
1598	x198	xx18	xxx1
2598	x298	xx28	xxx2
3598	x398	xx38	xxx3
4598	x498	xx48	xxx4
5598	x598	xx58	xxx5
6598	x698	xx68	xxx6
7598	x798	xx78	xxx7

- Neighbors of *Id* are chosen as to have the suffix of their identifier in common with *Id*

Pastry III

Pros

- $ln(n)$ messages guarantee
- Good path redundancy

Cons

- Difficult to keep a synchronized neighbor table
- Problem of data redundancy
- No adaptation to data dynamicity

Current state of Peer to Peer systems

- A lot of redundant systems
- Typically File Sharing

Common basic component

Distributed index (Key, Value)

- Key is typically the filename
- Value is typically the file content or where to obtain it

Each Key is associated with a node

Generic Interface

- Node Id : k-bit identifier (unique)
- Key : k-bit identifier (unique)
- Value : bytes (can be a file, an IP, ...)

Generic DHT (Distributed Hash Table)

- put(key, value)
  - Stores (key, value) on the node responsible of *key*
- value = get(key)
  - Retrieves the data associated with *key*

Current implementations

Software

- Kadmelia
- Chord
- CAN

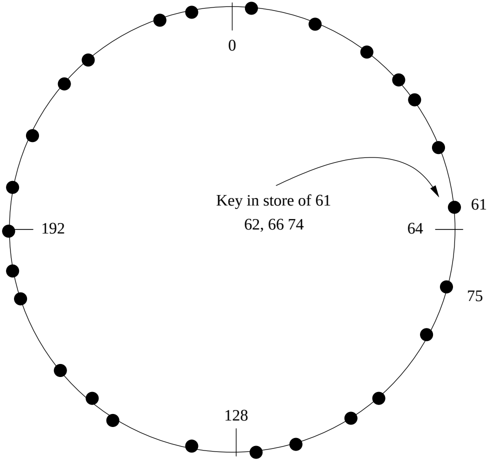
- Usage
  - File sharing
  - Naming
  - Chat service
  - Databases

Still limited

- Fundamental Problems
  - Complex request
  - Data coherence
  - Request with several answer
- Implementation difficulties
  - Distribute workload evenly
    - Keys
    - Requests
  - Only local information
  - Dynamic information

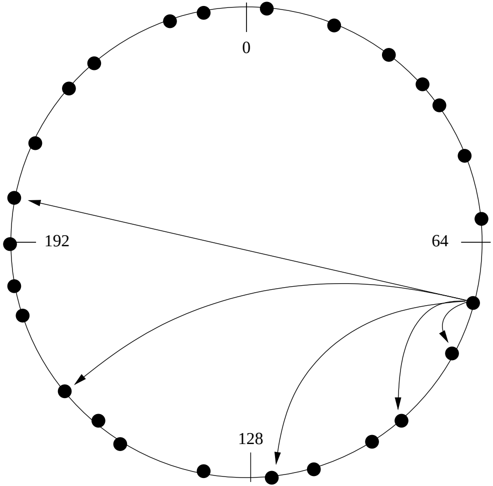
Chord structure

- Nodes are distributed on a circle
- Keys are assigned to the node with *Id* just before their value



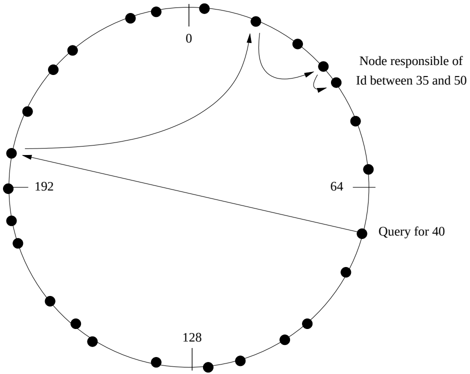
Neighbors

- $Log(N)$ neighbors
- Neighbors are nodes $Id+1$, $Id+2$, $Id+4$, ..., $Id+2^i$, ..., $Id+2^{k-1}$ (modulo 2^k).



Routing algorithm

- Forward to the neighbor which is prior to the key
- Query needs at most $Log(N)$ messages



Chord characteristics

Efficient

- If a (key, value) exists, the query will find it
- Fast : $Log_2(1.000.000) = 23$
- Small neighbors table $Log_2(N)$

Chord characteristics

Some problems

- Security and privacy
- Attack
- How to test and evaluate such system ?
- Real performance (instead of number of messages)

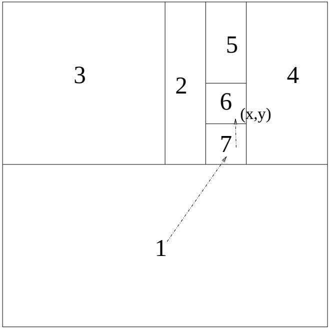
CAN (Content Adressable Networks)

- University of Berkeley
- Distributed Hash Table
- Basic operations : Insertion, lookup and destruction of couples (key, value)
- Cot de gestion indpendant du nombre de sites

CAN : structure

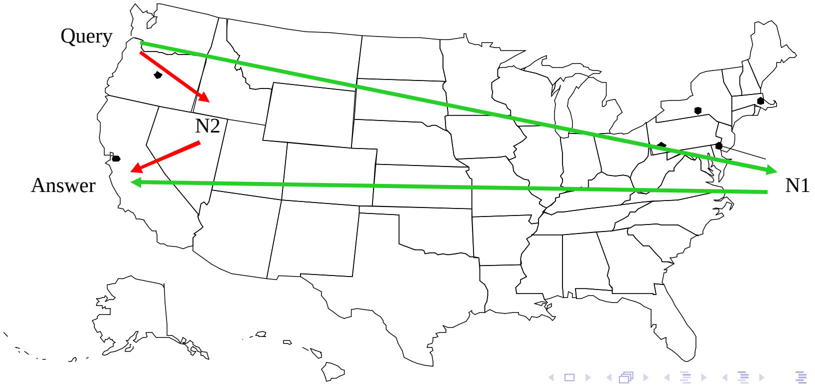
- Cartesian torus of Dimension d
- Each node manages a sub-space
- Keys are associated with points

- Simple routing
- Mean distance $= O(n^{1/d})$ (for n zones/nodes)
- Fault tolerant
- Supports churn



Physical overlay

- Logical topology mapped in the physical network :



Conclusion

- Peer to Peer systems are efficient for several uses (using border resources)
- Recent systems are scalable
- Low cost alternative to Client/Server
- Field old enough to be used in real cases
- Still not perfect
  - Trust & certification
  - Anonymity
  - Security
  - Performance
  - Layers fees

When to use Peer to Peer systems

- Limited budget
- Large audience
- Trusted users
- Dynamic system, but not too much
- Do not need guarantee
- Do not need control

Vision of the future

User centered

No more servers
All content provided and served by users

- Only cooperation of peers
  - Wikipedia
  - Social networks
  - Youtube
  - *Good Ol' Time* web-pages

Questions ?