P2P Techniques for Decentralized Applications

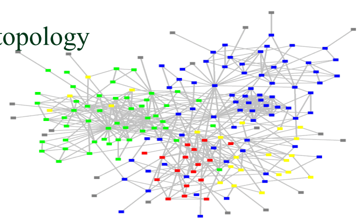
Esther Pacitti

Collaborators: R. Akbarinia, M.El-Dick, B. Kemme, F. Draidi

Context & objective

- Online communities
  - Users have common interests and are willing to share data
  - Dynamic and large scale (web scale) collaboration of members

Social network topology



CONTEXTS	File Sharing Collab. editing P2P CDN Social networks Recommendation, etc.

- P2P as light-weight alternative to centralized servers
- We mainly focus in the distributed architecture approaches

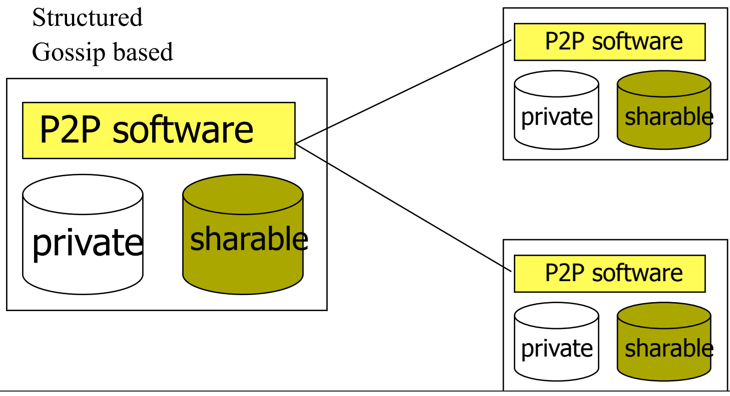
Outline

- P2P systems
- P2P Content Distribution Network (CDN)
- P2P Recommendation
- P2P collaborative editing using DHT’s
- Conclusion and Perspectives

3

P2P systems

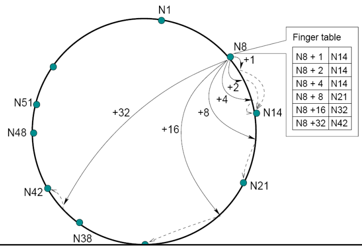
- Each peer can have the same functionality
- Decentralized control, large scale
- Main overlays:
  - non- structured
  - Structured
  - Gossip based



4

Structured overlay: Distributed Hash Table (DHT)

- Implements a hash table indexing functionality over P2P
  - Each peer *n* has an identifier *p* (hash(IP))
  - Each peer identified by *n* is responsable for a range of keys
  - Each peer is placed in the ring in ascending order
  - Notion of sucsessors and predecessors using a *finger table of m entries.*
  - Each entry points to specific peer *q*



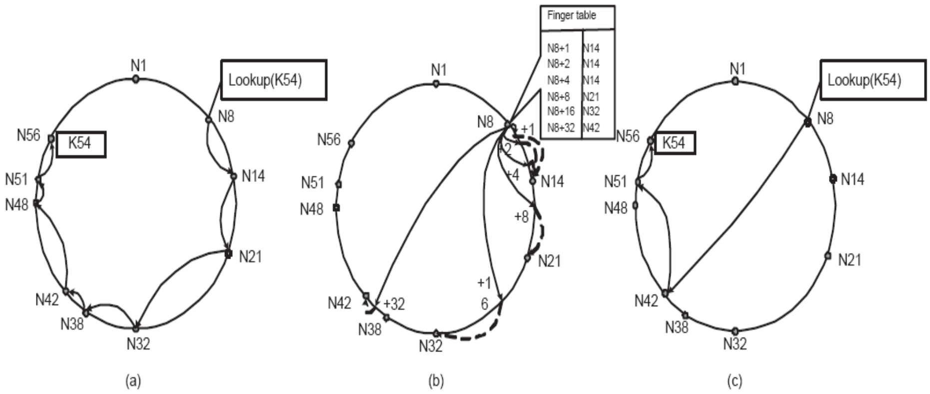
5

Structured overlay: Distributed Hash Table (DHT)

- Deterministic: maps a given *key* onto *p*, called *responsible of key*, and looks up *p* efficiently using *finger tables*
- *key = hash (filename)*
- *Lookup(key) -> p*
- *put(key, value)*: stores a pair *(key, value)* at the peer that is responsible for *key*
- *get(key)*: retrieves the value associated with *key*
- Complexity (O(log(n))
- Scalability, efficient data location
- No freedom for data placement
- Examples: Can, Chord, Pastry, etc

6

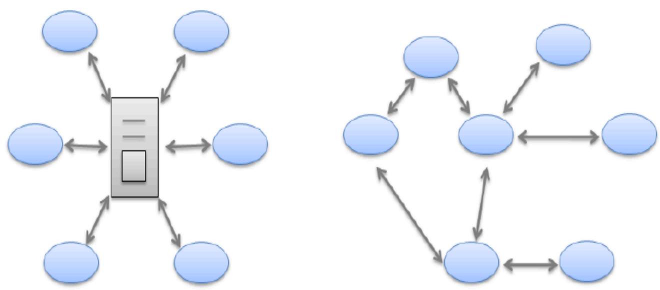
Structured overlay: Distributed Hash Table (DHT)



Structured overlay: Distributed Hash Table (DHT)

- Churn Behavior
- Arrivals
  - New peer gets the responsabilities of some of its sucessor keys
  - The DHT stabilization layer updates the involved finger tables
- Departs
  - The departing peer transfers the responsabilities of his keys to sucessor
  - The DHT stabilization layer updates the involved finger tables
- Failures
  - Peers detects the neighbors failures and triggers the DHT stabilization layer to update the involved peers
- Not well suited for high degree of churns
  - Stabilization layer does not provide guarantess

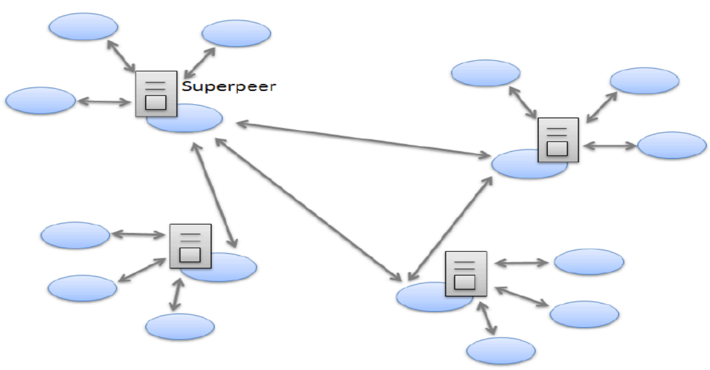
Unstructured Overlays



(a) Centralized. (b) Pure decentralized.

Unstructured Overlays

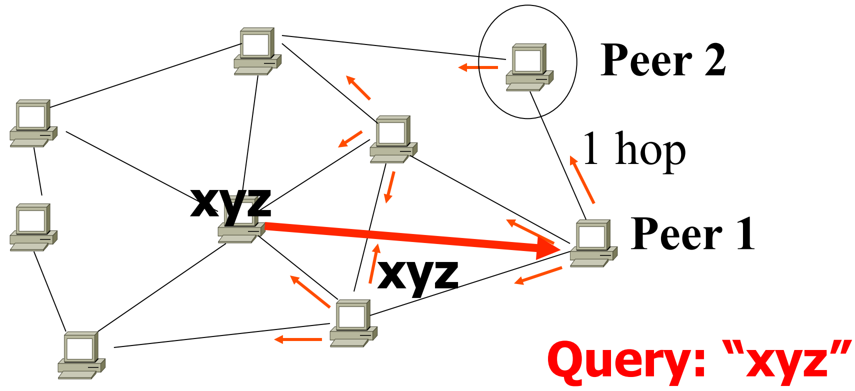
Each peer has a established link with some neighbors



(c) Partially decentralized with superpeers.

Decentralized Unstructured Overlays

Gnutella: flooding approach



11

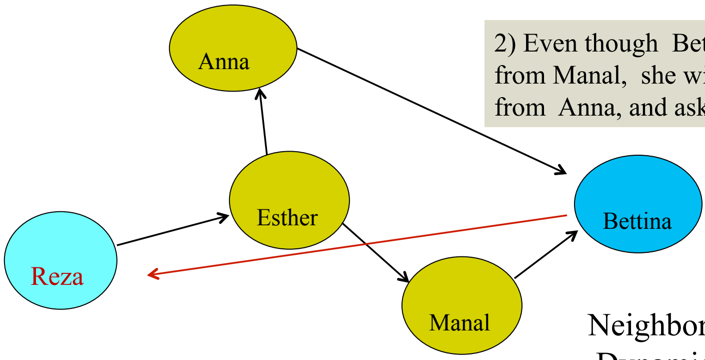
Unstructured Overlays

- Only neighbors id’s are needed for query routing: non-deterministic
- Query routing (flooding)
  - blindness and redundancy
- In average any two peers are less than 7 hops away
- Natural replication => fault tolerance, robustness
- Data is stored locally
- Well suited for churn
  - No stabilization layer
- Generates heavy network traffic
- A query necessitates *O(n)* hops to find a object
- Other protocols related protocols:
  - breadth-first-search, iterative deepening, random walk

12

Gossip based overlays (Dynamic Overlays)

2) Even though Bettina does not learn from Manal, she will probably learn from Anna, and ask Reza for a ticket



Neighbors links changes Dynamically

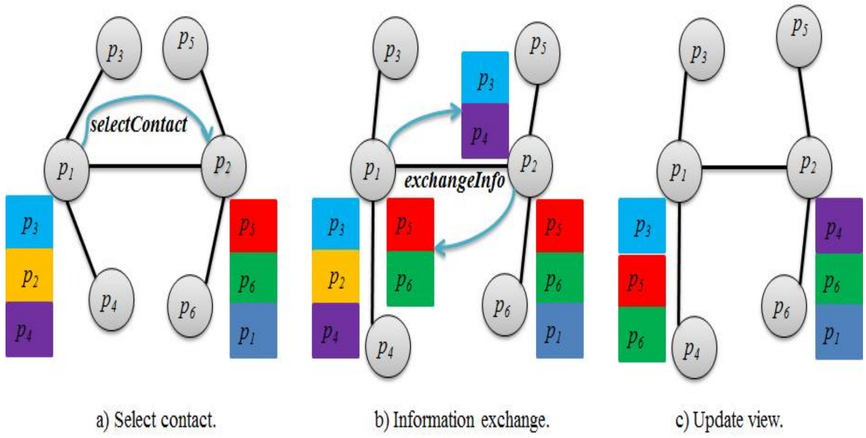
1) Reza tells Esther: I have tickets to the show

- *Background data dissemination protocols* continuously gossip about information associated with the participating nodes

Gossip-based Overlays

- Widely used in P2P systems
  - Overlay construction, information dissemination, data replication, etc
- Main approach: each peer periodically exchanges its information with another randomly-selected peer
- Each peer keeps locally a view of
  - its dynamic acquaintances (or view entries)
- At each peer the behavior of gossip protocols is modelled:
  - Active behavior: how to initiate to initiate a gossip exchange
  - Passive behavior: how to react to a gossip exchange
- Gossip protocols consist of three modules:
  - SelectContact
  - ExchangesInfo
  - UpdateInfo

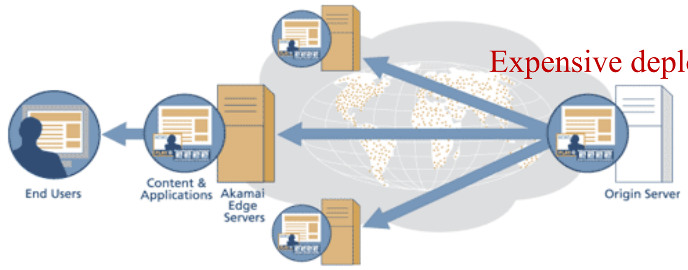
Gossip: example



15

Context: Content Distribution Networks

- A typical commercial CDN (e.g. Akamai)
  - Sits between content providers and content consumers
  - Has hundreds of servers throughout Internet
  - Replicates its customers' content in CDN servers
  - Updates servers when provider updates content



16

Flower-CDN*

To enable websites of non-profit organizations, to efficiently distribute its content with help of the community interested in its content

- Replace *n* edge servers by *m volunteers peers (m >> n)* that are willing to collaborate in a same *interest* (website)
- Locality aware P2P CDN
  - Search for a content as close as possible
- Key idea: *clients keep their requested content to serve it for others*
- Approach: combine DHT efficiency and gossip robustness
  - Hybrid P2P overlay

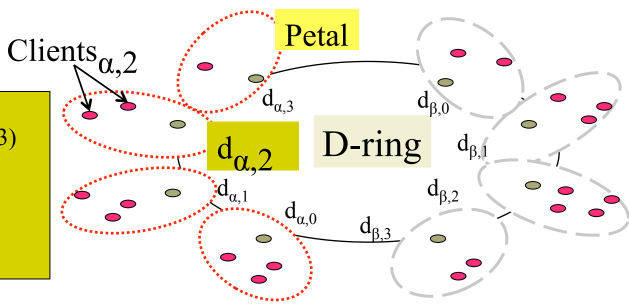
*M. El Dick, E. Pacitti, B. Kemme. Flower-CDN: a Hybrid P2P Overlay for Efficient Query Processing in CDN. *Int. Conf. on Extending Database Technology (EDBT),* 2009.

Flower-CDN architecture

- Hybrid and locality-aware
  - DHT-based overlay (D-ring) that serves as a P2P directory service
  - Gossip-based overlay (Petals)
    - Clusters clients that share interest in a website *ws* wrt a locality *loc*.

Websites={α, β}
4 localities (0...3)

$d_{\alpha,2}$ = directory peer for α in locality 2



Flower-CDN

- Each D-ring peer
  - P2P Directory service based on DHT lookup service
  - Provides efficient access to a petal for new clients
  - Provides directory information wrt to its neighbors in the ring to help query handling if necessary
    - Directory peers of *a website* are neighbors in D-ring
- Within a petal
  - Dynamic overlay for content search
  - Serves queries on behalf of a *website* wrt a *locality*
  - Clients share contents (popular transfered pages)
  - Query search is done by gossiping
    - Storage and exchange of popular content of *ws*

D-ring: P2P directory

Structured overlay with novel DHT mechanism
Construction based on peers' interests and localities

Peer Id is split into 2 segments: hash(website ID) + locality ID

website ID	locality ID

hash (url_{ws}) [0 .. k-1]

Each directory peer is reponsible for 1 key

Each website *ws* is covered by *k* directory peers wrt to localities (landmark-based techniques [Ratnasamy 02])

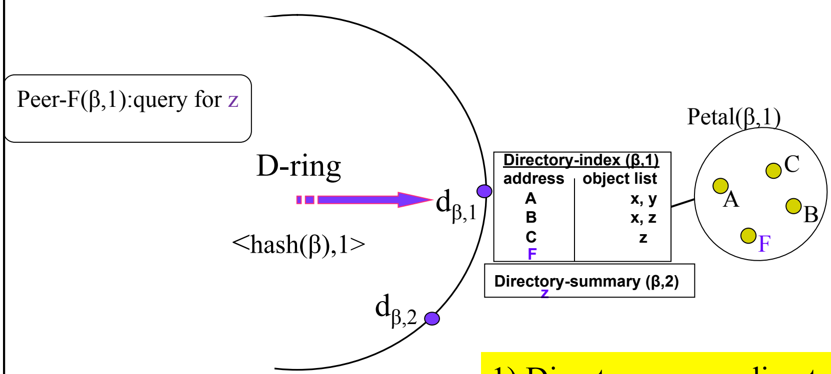
D-ring lookup service

Using standard DHT lookup service

- Case of a query request for a *content* wrt to *website* from a client at locality *loc*
  - key = hash(*website-id*) + *loc*
  - Lookup(key) -> directory peer wrt to the *website and locality*
    - *Search for content in the corresponding petal*
  - If the directory peer for that *locality* does not exist, then *client* becomes a new directory peer

21

D-ring query processing



D-ring = 1st access to content overlays

1) Directory peer redirects the query to the peer that might hold z
2) If z is not found, the query is re-directed to a neighbor directory

22

22

Petals

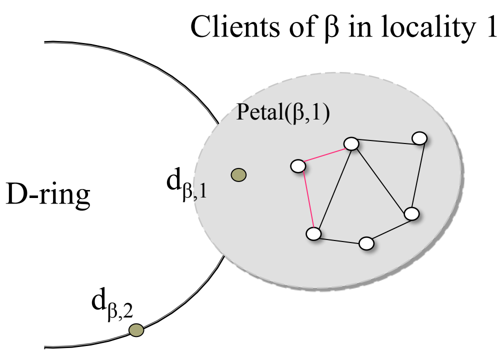
Clustering peers according to their interest and their locality
$Petal(ws,loc) = d_{ws,loc} + \{client_{ws,loc}\}$

1. After being served, client becomes $client_{ws,loc}$
2. Each client holds a view of its petal:
   - group of contacts known by the client (includes content summary info)
3. Periodically, each client selects a contact from its view to gossip (Cyclon: [Voulgaris et. al 05]) and update its view

Clients in a petal gossip to spread information about the content they have

23

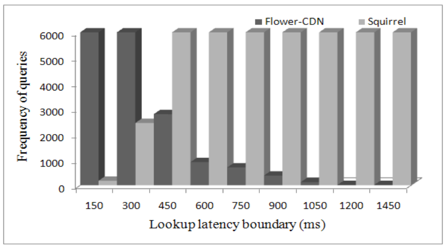
Petals

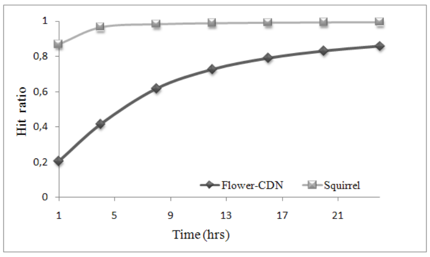


1) Join via D-ring

2) Get subset of contacts from someone in Petal(β,1)
3) Periodic gossip exchanges summary info in Petal (β,1) to update client view
4) By checking its local summary info, a client may see where a copy of the requested object might be stored

Performance Results





- Evaluation compared with an important P2P-CDN solution (Squirrel: [Sitaram et. Al. 02], based on pure DHT) :
  - Flower-CDN reduces lookup latency by a factor of 9 with slight decrease on hit ratio
  - Flower-CDN reduces transfer distance by a factor of 2

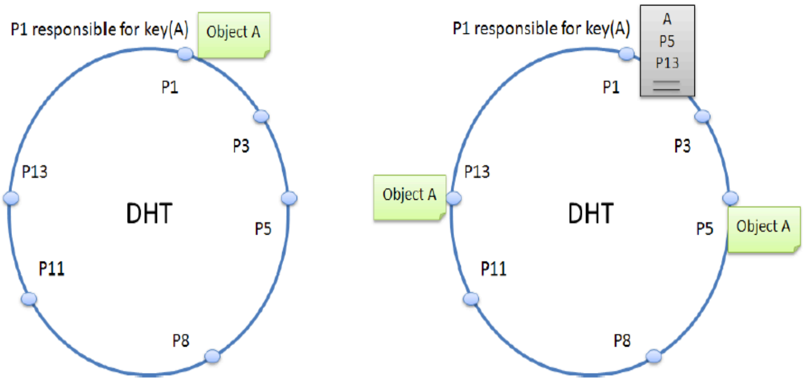
25

Related Approaches

- Unstructured Approach
- Poofs: Peers keep requested objects and can then provide them to other participants. To locate one of the object replicas, a query is flooded to a random subset, of neighbors with a *fixed time-to-live (TTL) i.e., the max number of hops*
- Structured Approach: Squirrel
- Home-based: It places objects at peers with ID numerically closest to the hash of the URL of the object without any locality or interest considerations (see Figure1.19a). Queries find the peer that has the object by navigating through the DHT.
- Directory-based: stores at the peer identified by the hash of the object's URL a small directory of pointers to recent downloaders of the object

26

Structured Approach: Squirrel



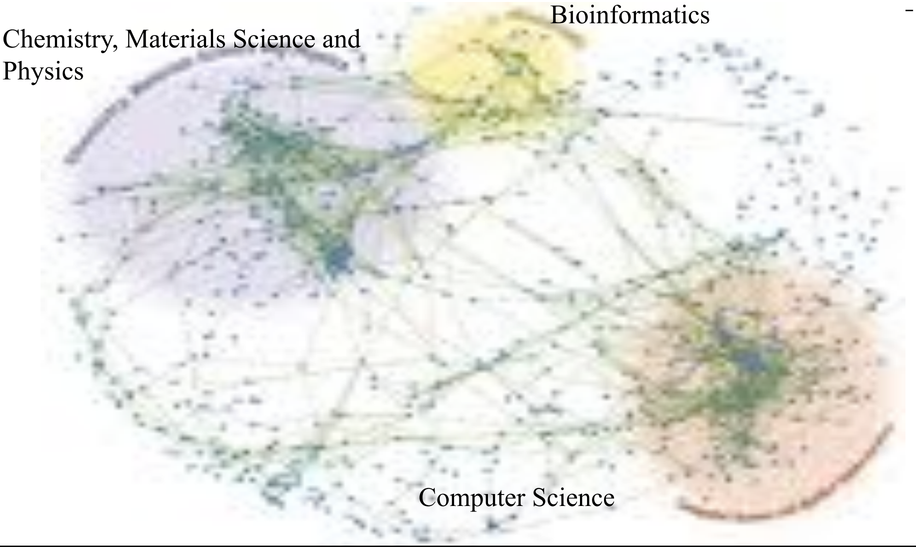
(a) Home model.

(b) Directory model.

Perspactives

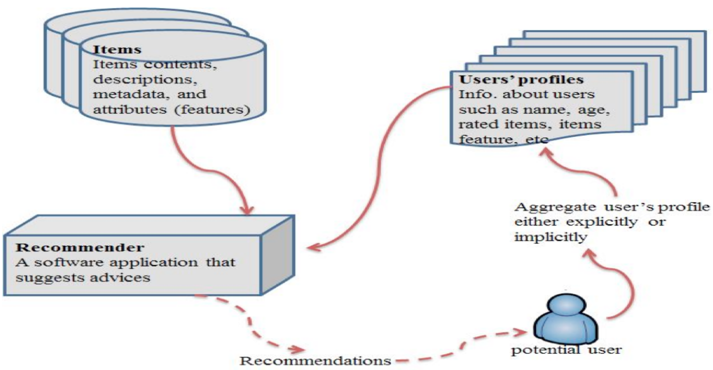
- Introduce some personalization on content sharing
  - Similar users issue similar queries -> store similar contents
  - Exploit similar users contents
  - Become friend with similar users - > Social Networks
  - Provide Recommendation

Motivation for P2P Recommendation



Recommendation in the Web

- Helps to choose among a large range of alternatives by exploiting historical patterns.



30

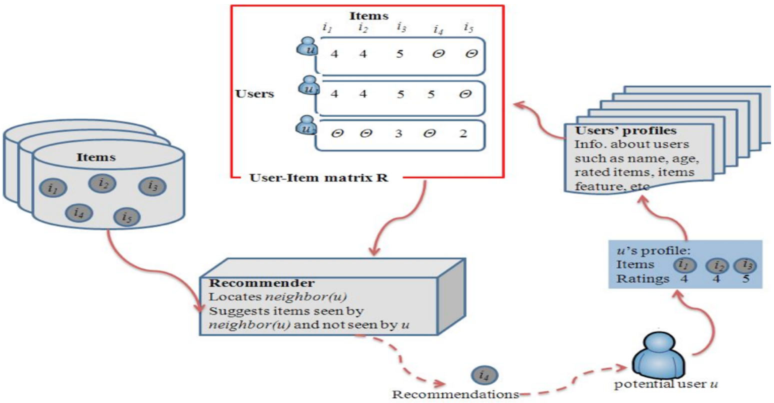
Collaborative Filtering

- Recommends to *u items (photos, links, etc) that have been rated by users who share similar interests based on*
  - ***tagging** or **rating** behavior*
- Main steps:
  - Mesure the similarity between a user *u* asking for recommendation and all users in the system
  - Select those users who are most similar to *u* that become neighbors of *u*
  - Predict missing rates
  - Provide recommendation based on *u* neighbors based on a Top-k approach

31

Collaborative Filtering

- Measures the similarity between *u and all users in the system*
  - recommendation is done using a matrix model: space consuming

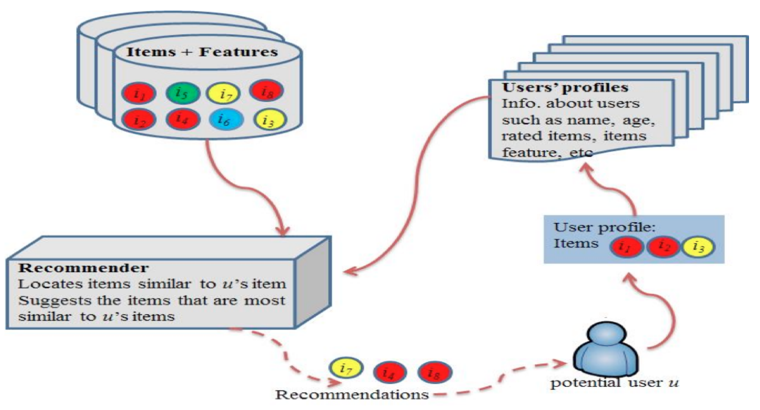


32

Content based Recommendation

- Recommend to a user *u items that are similar to u's previously rated items or similar to the user profile*
- Contents are indexed, for instance, by key words (e.g. TF-IDF metric)
- User profile (topics of intrest) are derived based on his content
- Main Steps
  - Compute user *u* topics of intrest (vector of keywords)
  - Measure the similarity bewteen *u* and and each content that *u* did not see or rate yet (based on vector-space methods: cosine similarity)
  - Select the most similar contents
- User is limited to receive products that are only similar to those it has rated

Content based Recommendation



Recommendation based of friendship

- Improves the quality of recommendation
  - Similar trusful friends are good recommendors
- Modeled as a graph
- Avoids the Cold Start Problem
- Exploits trust networks
- User tags are used to measure users similarities and similarity bewtween items
- *Small-world phenomena*
  - A user can contact any other user in few hops
  - Enables efficiency

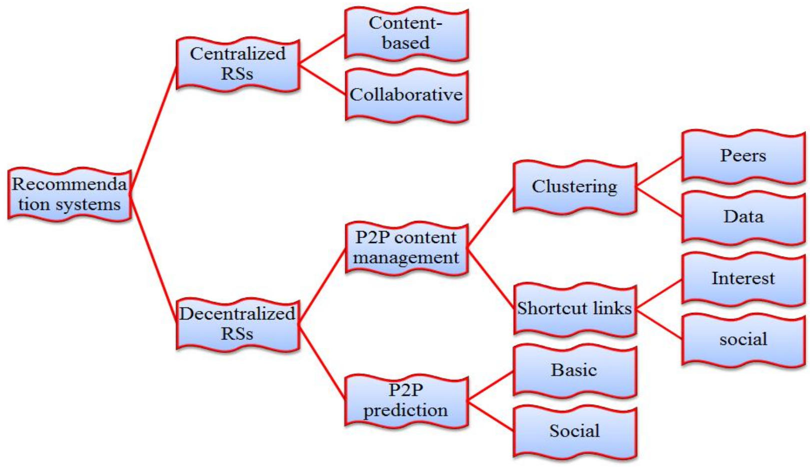
P2P Recommendation: Content Management Systems

- Used to build distributed information retrieval systems (e.g.key-word queries in Google)
- Clustering overlay:
  - Cluster similar peers based on the contents they store, or
  - Stores similar contents in a same peer
  - Examples: [Hai,WWW06] [SETS: Bawa,SIGIR'03], [Garcia-Molina 03]
- Shortcut links: peers establishes direct links with other peers who are similar wrt to intrest or social behavior.
  - Examples: [SPOUT '04], [P4Q'10], [TRIBLE '08]

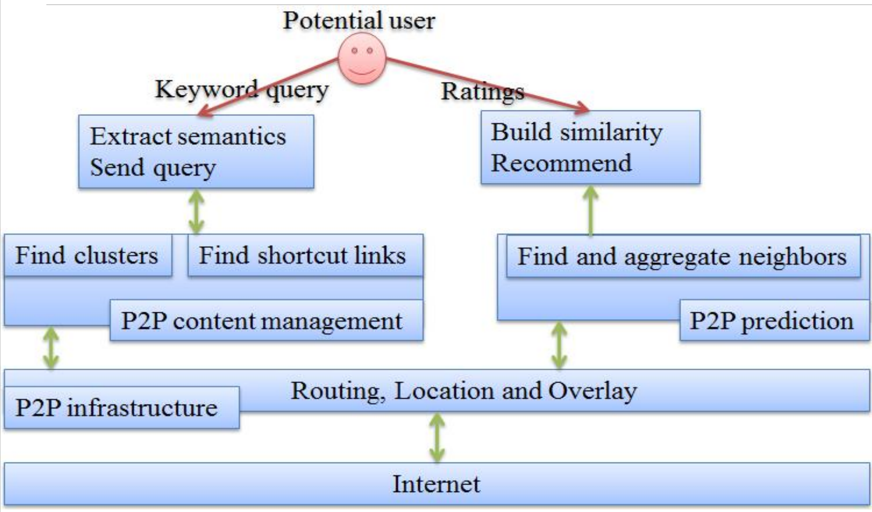
P2P Recommendation: Prediction Systems

- Basic prediction: based only in users rates
  - Examples: [Tveit'01], [PocketLens'04], [Kermerrec, OPODIS, 10]
- Social P2P prediction: laverages users preferences (rating) with users social data (friends, trust, etc)
  - Examples: [Kim'03], [Goldneck'06], [Kruk'06]

Recommendation Systems Approaches



P2P Recommendation



39

P2Prec: recommendation for on-line communities*

- Social Prediction approach that exploits shortcut links
- We exploit the fact that people tend to store content related to their **topics of interests**
- Users' topics of interest can be automatically derived from the contents or documents they store and the ratings they give, without requiring tagging

Draidi, F., Pacitti, E., Kemme, B., P2Prec: a P2P Recommendation System for Large-scale Data Sharing. *Tran. on Large-Scale Data- and Knowledge- Centered Systems, LNCS, 6790(3):87-116, 2011*

40

Recommendation Model

- *D is the set of shared rated documents* $doc_i \ldots doc_n$
- *U is the set of users in the system, corresponding to autonomous peers*
- *Topic management**
  - *T is the set of global topics*
  - $T_u \subset T$, *is the set of users' topics of interest (based on rating and relative nº of documents)*
  - $T_u^r \subset T_u$ *is the set of users' relevant topics (based on rating and the absolute nº of documents on the topic)*
- *Q: key-word queries that are mapped to topics* $T_q \subset T$

To anwser a query we rely on relevant users wrt to T_q

*Automatically extracted using LDA (used in IR)

41

Recommendation Model

- Recommendation is based on rates and popularity
- Query answer to q
  - $recommendation_q = rank(rec_{q1}(doc_i), \ldots rec_{qn}(doc_j))$

Where

$rank(rec_{qi}(doc)) = a*similarity(doc,q) + b*pop(doc)$

Problems:
How to find relevant users wrt to a query ?
How to trust the recommendation ?

42

Finding Relevant Users: Gossip Approach

-Disseminate relevant users information by gossiping
-Gossip view is dynamically updated
-In the event of a query at *u*

-u *searches for similar relevant users v∈u's local-view so that v can give recommendation for q.*

-at each selected *v*, the gossip view is recursively exploited to serve the query, until TTL.

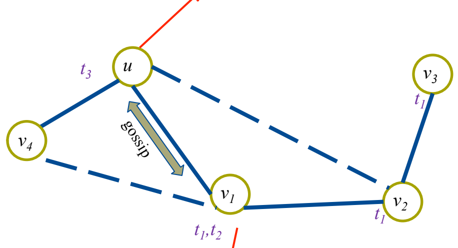
-u receives recommendations and ranks them

43

Relevant Users Dissemination with Random Gossip

u's *local-view* after gossip

User	Gossip information
v_1	t_1, t_2
v_4	t_2,t_3
v_2	t_1



v_1's *local-view* after gossip

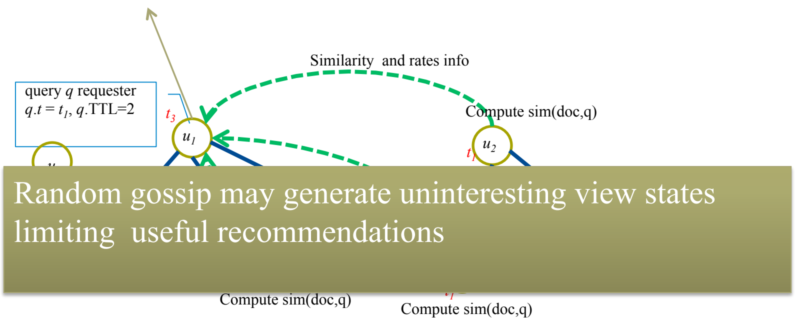
User	Gossip information
v_2	t_1

1) Each user *u* maintains a *local-view*
2) Each user *u* periodically selects
   – a random contact *v* to gossip with
   – a gossip message and send it to *v*
3) Each user *u* recieves a gossip message
   – Updates its *local-view*

44

Query Processing

$$rank(doc) = a * sim(doc,q) + b * pop(doc)$$



Random gossip may generate uninteresting view states limiting useful recommendations

1) Query q is mapped to topics T_q
2) Select Top-k contacts in the gossip view wrt to the query topics (cosine similarity)
3) Redirect Query
4) Do 2) and 3) Recursively until TTL

Semantic Gossiping

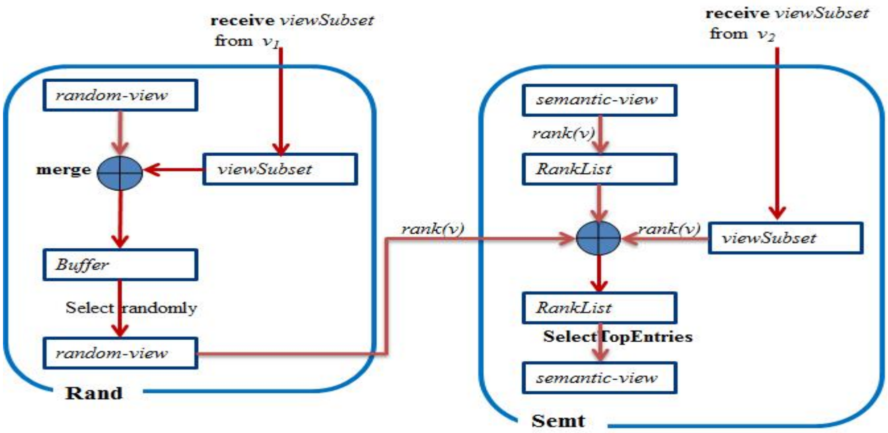
- Idea: user *u* maintains a local-view of relevant similar users with high hit-ratio
- Hit-ratio is defined as the percentage of the number of queries that have been answered sucessfully
  - When *u initiates a query q , it searches for relevant users v∈u's local-view so that v can give recommendation for q. If u finds such relevant users*
- However, Semantic Gossip may reduce the user's ability to discover new relevant users
- *u hit-ratio can be easily added as an attribute of a local-view entry, and becomes part of the gossip message*
- Gossip is done with one of the most similar relevant user that is similar to *u, with high hit-ratio,* not chosen recently to learn about new similar contacts
- Query processing is the same as before

Semantic Two-layered Gossip

- What about combining Random and Semantic Gossip ?
- Inspired on [Voulgaris10]
- In our approach the two views are managed asynchronously
- Random View: to find new useful contacts
- Semantic View: works as before
- Random view is also taken into account to update the semantic view

47

Semantic Two-layered Gossip



48

Open Issues: Trusted Recommendation

- Exchange high recall by trusted recommendation
- Exploit Friend to Friend recommendation instead of *anonymous* recommendation
- Define a trust model based on friendship and social structures
- Idea: keep all found relevant users found during random gossiping that are declared friends in a local file (FOAF file)
  - FOAF provides an open, detailed description of profiles of users and the relationships between them using a machine-readable syntax
- Use the FOAF file to serve queries instead of the gossip views
- New social metrics:
  - Similarity and trust among friends networks
  - Diversity (not only similar documents, or friends)
  - Exploit the popularity of a document as the number of replicas in P2Prec
- Define a metric to express user satisfaction

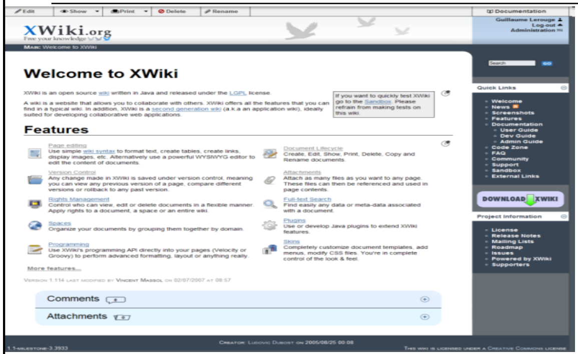
49

Trends

- Existing social model are not enough
- Define a Social Model suited for specific communities (e.g. scientific researchers) based on communities hierarchies, reputation, etc.
- Case study: Collaboration among differents research communities around the world that are willing to share phenotypage data to study plant behavior in different conditions (temperature, season, etc).
- Exploit Cloud facilities and several Cloud instances
  - Each peer stores his data in the cloud in a controled way

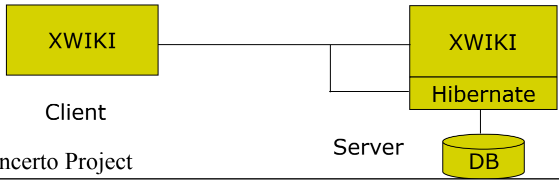
50

P2P collaborative editing using DHT's



Enterprise Wiki system in OSS from Xpertnet, Paris

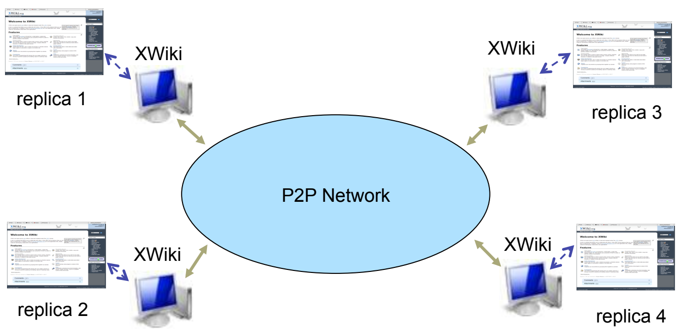
- Collaborative text editing among multiple users
- Wiki-page updating (last save wins)
- Client-server architecture



*ANR Xwiki Concerto Project

51

Xwiki Context



How to ensure eventual consistency ?

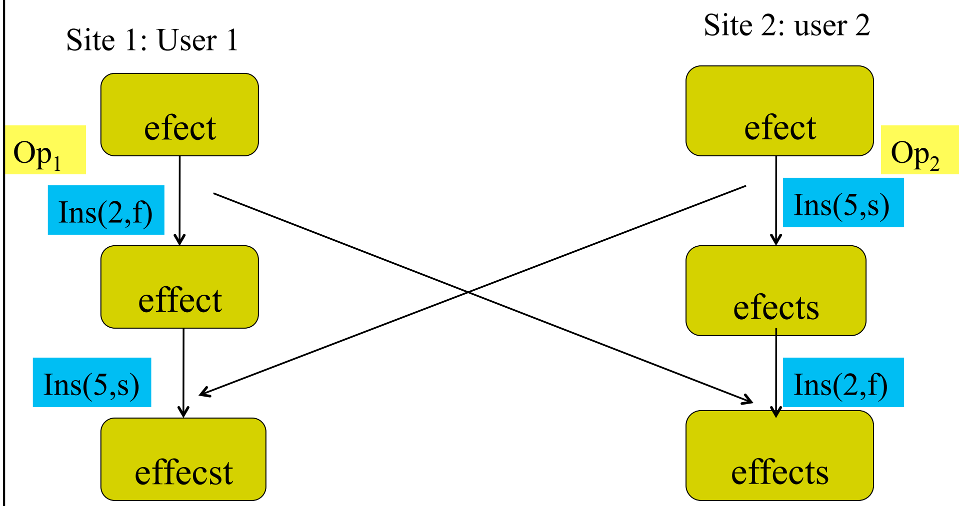
52

Approach

- Multi-master replication using *Operational Transforms (OT)* (Ferrié et al. [04 ,07], Molli et al. [07,08], etc)
  - Replicates a document in all sites and allows edition operation
    - Operations: insertion, delete, update
  - Remote operations are transformed before execution to repair inconsistencies, wrt to a specific order
- The challenge is to provide
  - Eventual consistency and scalability

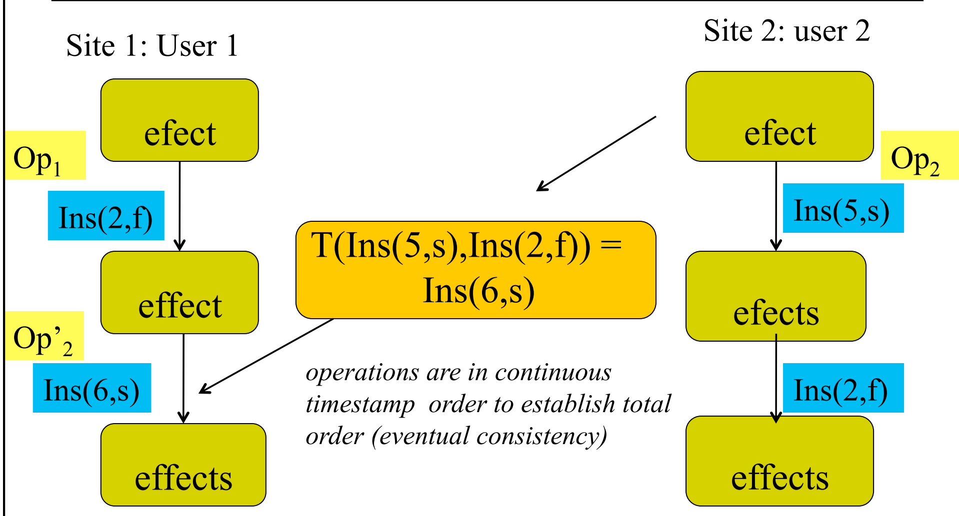
53

Example of inconsistency



54

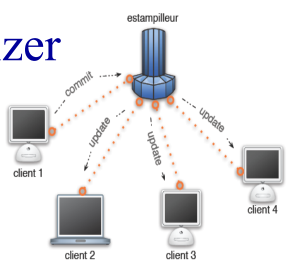
Reconciliation with SO6 [Molli et. al. 03]



OT with SO6*

- Each operation is broadcast to all users.
- All operations need be timestamped in continuous order (1,2,3…)
  - Enables concurrent users to be aware of how many remote operations they miss before applying the transformation

Synchronizer

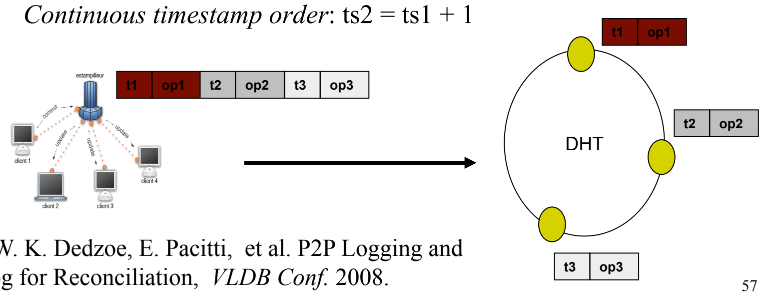


- Centralized timestamper
  - Limited scalability
  - Bottleneck
  - Single point of failure
  - Limited Log

* P. Molli, G. Oster, H. Skaf-Molli, A. Imine. Using the transformational approach to build a safe and Generic data synchronizer. *ACM SIGGROUP Conf.*, 2003.

P2P Logging and Timestaming for Reconciliation (P2P-LTR)

- Replaces the centralized synchronizer by P2P synchronizer
- Replaces the centralized log by P2P log
- Major functions
  - Logging of user operations (patchs) in a DHT
  - P2P continuous timestamping of these operations
    - *Continuous timestamp order*: ts2 = ts1 + 1



*M. Tlili, W. K. Dedzoe, E. Pacitti, et al. P2P Logging and Timestampg for Reconciliation, *VLDB Conf.* 2008.

57

P2P-LTR model

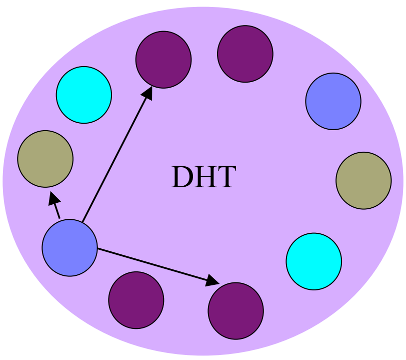
Each XWiki document is identified by a key

Xwiki peer: application

Master-key:
1) generates timestamps for a given document (provides last-ts)
2) publishes consistently patches in log peers

Master-key succ: replaces Master-key after crash

Log peer: stores timestamped patches for a key

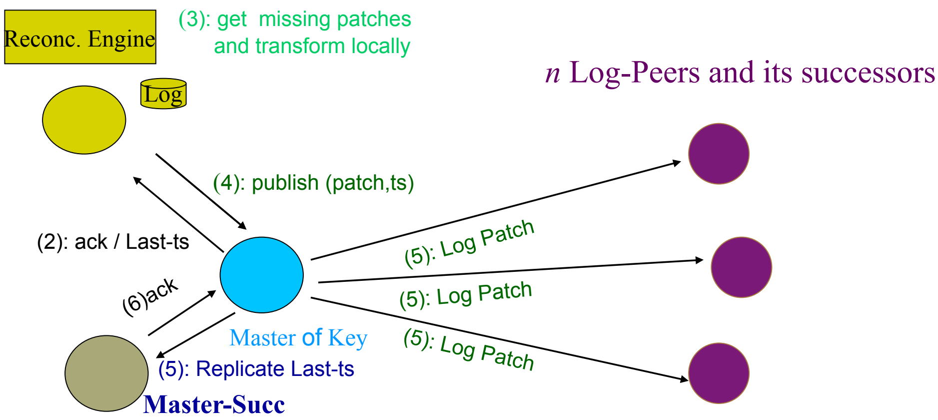


In the Internet or in a a Cloud

58

P2P-LTR reconciliation

Xwiki-Client: Post (key, patch, ts)



59

Results and Perspectives

- Performance
  - Response times are significantly improved (up to a factor of 3) regardless of the number of documents
  - Low impact of failures, on response time:
    - E.g. by increasing the failure rate by 5, the response time increases by about 11%
- Wide applicability
  - Data storage in the Cloud

60

Conclusion: P2P-> Cloud2Cloud (C2C)

- P2P are good for privacy reasons
  - Avoids storing data in remote untrusted infratrusuctures that may not scale well
- However P2P are vunerable to attacks
- Clouds may offer more gurantees and storage facilities (NoSql, elasticity, etc)
- What about exploiting private C2C architectures for decentralization ?

61

Thanks !

Questions ?

62