Preferences and Games for Sharing Distributed Content

Brief Research Overview of DMOD network-centric group

Sept 09@Georgia Tech

Evaggelia Pitoura
Department of Computer Science
University of Ioannina, Greece

http://dmod.cs.uoi.gr

An overview talk on our current research

on making content sharing in distributed systems efficient and effective.

Current focus on

- Preferences and Ranking
- Overlay Networks

Be Aware: Overview -> may switch from topic to topic faster than expected

Part 1: PREFERENCES
Part 2: OVERLAYS

Why Preferences?

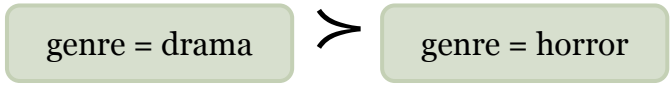
Huge amount of information available to diverse population => need to personalize

How?

Through preferences: Two fundamental philosophies for expressing preferences:

- Quantitative approach
  - Using scoring functions that assign a numeric score (interest) to an item:

- Qualitative approach
  - Binary relations between pairs of items:



Our Work on Preferences

- Preferences and Keyword Search in Relational Databases [under submission]
- **Y**ou **M**ay **A**lso **L**ike Results (**YMAL**) in Databases [ongoing – check out our workshop paper at **PersDB09**]
- Context-aware preference models [**ICDE07**, **EDBT08**] (2 slides)
  - extend preference models with a context component
- Preferential Publish/Subscribe [**DEBS09**] (a bit more)
  - Introduce ranking (top-*k* delivery) in pub/sub

Joint work with

Kostas Stefanidis http://www.cs.uoi.gr/~kstef

Marina Drosou http://www.cs.uoi.gr/~mdrosou

Context-Aware Preferences in a nutshell

Preference with two parts

(context-descriptor, preference)

context

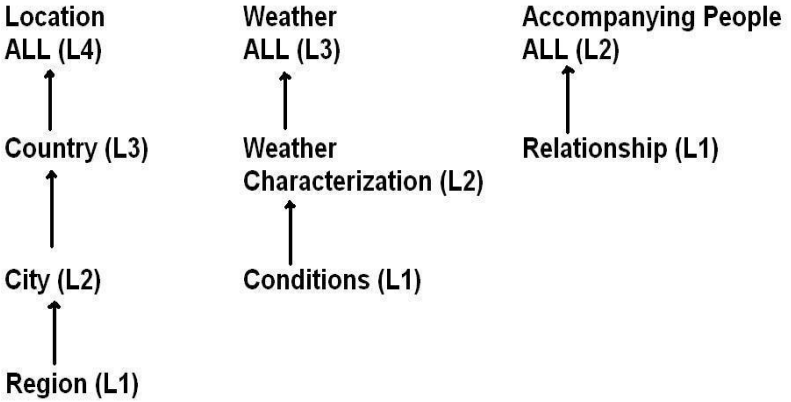
- While in GT with the DiSL people, I like to talk about research
- While in Greece with friends, I like to talk about music
- In Ioannina (Greece), gastra
- In Lyon (France), quannelle

Context-Aware Preferences in a nutshell

- Context Attributes
- Hierarchical Domains
- Context Resolution:

Find the preferences applicable to the current context

through a special data structure the Profile Tree



- **Efficient top-*k* computation through pre-computation of representatives rankings (representatives defined through clustering preferences)**

Our Work on Preferences

- Preferences and Keyword Search in Relational Databases [under submission]
- **Y**ou **M**ay **A**lso **L**ike Results (**YMAL**) in Databases [ongoing – check out our workshop paper at PersDB09]
- Context-aware preference models [ICDE07, EDBT08]
  - extend preference models with a context component
- Preferential Publish/Subscribe [DEBS09]
  - Introduce ranking (top-*k* delivery) in pub/sub

Publish/Subcribe Systems

Publish/Subscribe offers an attractive alternative to typical searching

Users do not need to repeatedly search for new interesting data

They specify their interests *once* and the system automatically notifies them whenever relevant data is made available

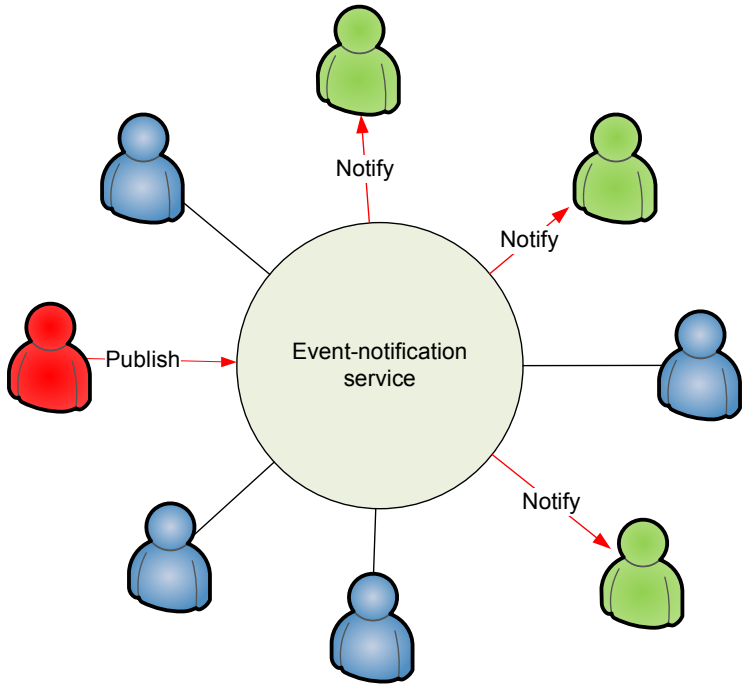
Examples:

- Google Alerts
- Twitter
- Microsoft BizTalk Server

Pub/Sub Systems

Parts of a Publish/Subscribe system:

- Subscribers: consumers of events
  - enter subscriptions + receive notification
- Publishers: generators of events
- Event-notification service:
  - Store subscriptions (user interests)
  - Match events with subscriptions
  - Deliver event notifications *to interested subscribers only*



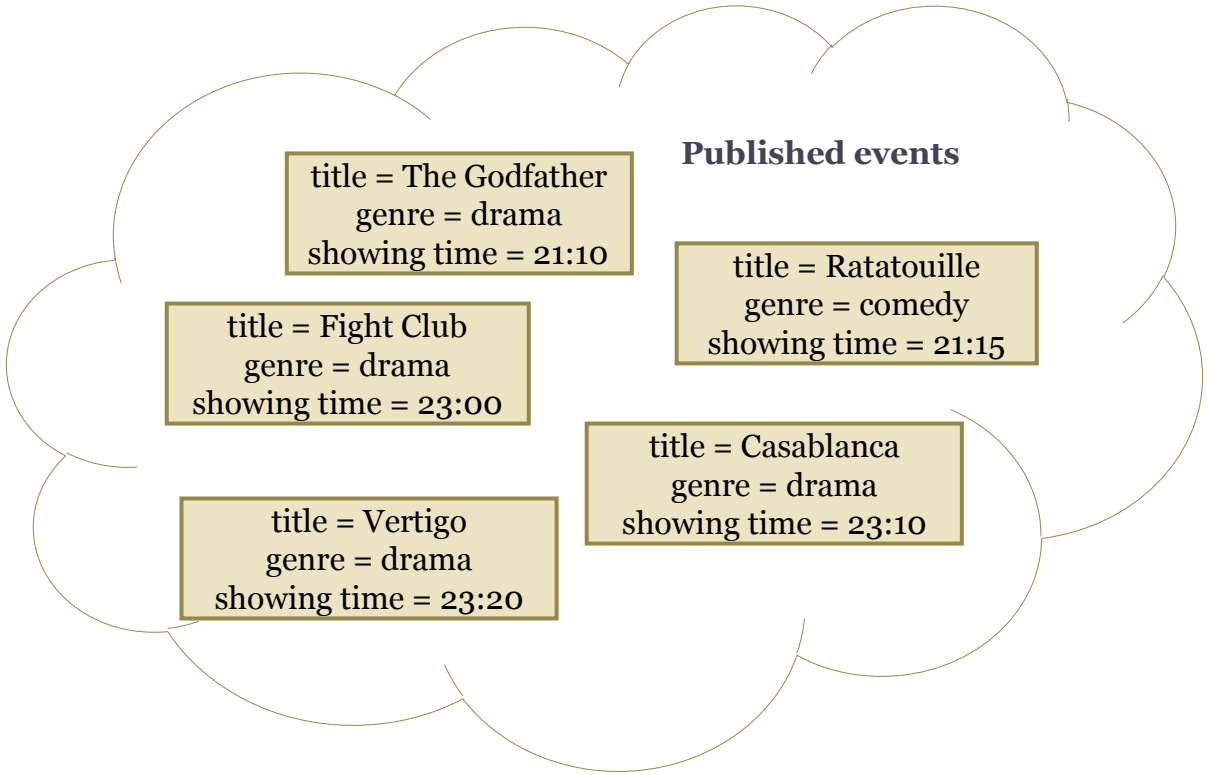
Topic-based

- Each event belongs to a number of topics (e.g. “music”, “sport”)
- Users subscribe to topics and receive all relevant events

Content-based

- Users subscribe to the actual content of the events, or a structured summary of it

Pub/Sub



User subscriptions

genre = drama

genre = horror

Why Preferential Pub/Sub?

Typically, all subscriptions are considered equally important

But, users may receive:

- overwhelming amounts of notifications
- too much overlapping information

- In such cases, users would like to receive only a fraction of notifications, the most interesting ones:
  - Say Addison is more interested in horror movies than comedies
  - Addison would like to receive notifications about (various) comedies *only if there are no (or just a few)* notifications about horror movies => ranking

  - Current publish/subscribe systems do not allow Addison to express this different degree of interest

Preferential Pub/Sub

Extend subscription with preferences: users define *priorities* or degrees of interest on their subscriptions

preferential subscriptions

Use preferential subscriptions, we deliver to users only the *k* most interesting (highly ranked) events

Preferential Subscriptions

We focus on a qualitative model (quantitative can also be used):

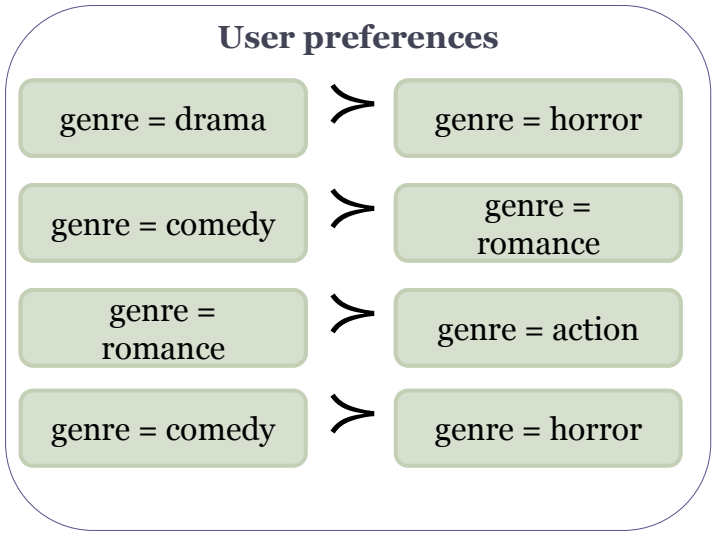
- More expressive
- More intuitive

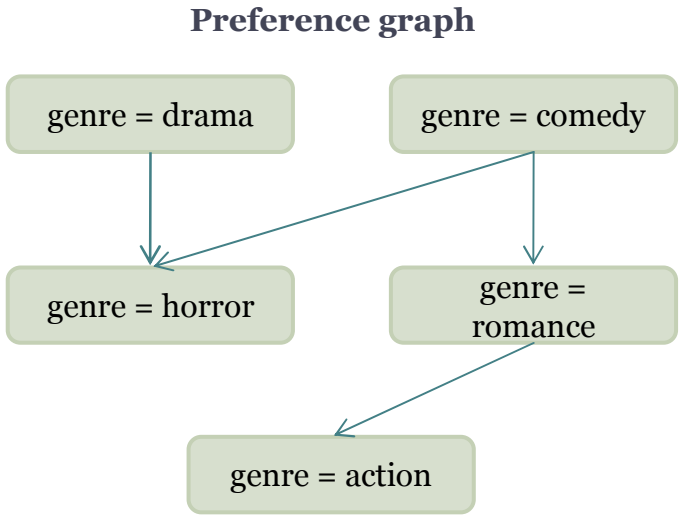
PREFERENTIAL SUBSCRIPTION MODEL

Let S^X be the set of subscriptions of user X. Along with S^X, X specifies a binary preference relation C^X on S^X, $C^X = \{(\boldsymbol{s_i} \succ \boldsymbol{s_j}) \mid s_i, s_j \in S^X\}$, where $s_i \succ s_j$ denotes that X prefers s_i over s_j or considers s_i more interesting than s_j.

Preferential Pub/Sub

We organize subscriptions in a DAG:



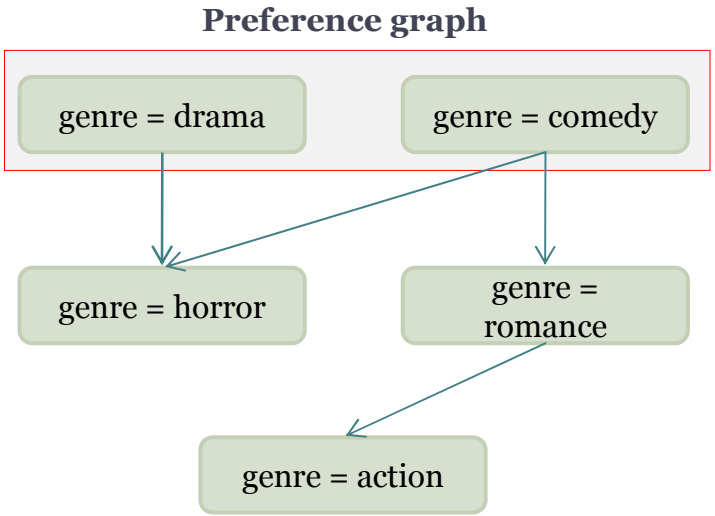


Preferential Pub/Sub

We perform a topological sort to compute winnow levels. The subscriptions of level i are associated with a preference rank $\mathcal{G}(i)$:

- $\mathcal{G}$ is a monotonically decreasing function with $\mathcal{G} \rightarrow [0, 1]$

e.g. for $\mathcal{G} = (D+1-(I\text{-}1)) / (D+I)$



Preference rank = 1

Preference rank = 2/3

Preference rank = 1/3

Preferential Pub/Sub

Given a set of user preferential subscriptions and a published event:

- Find out how important the notification is to the user

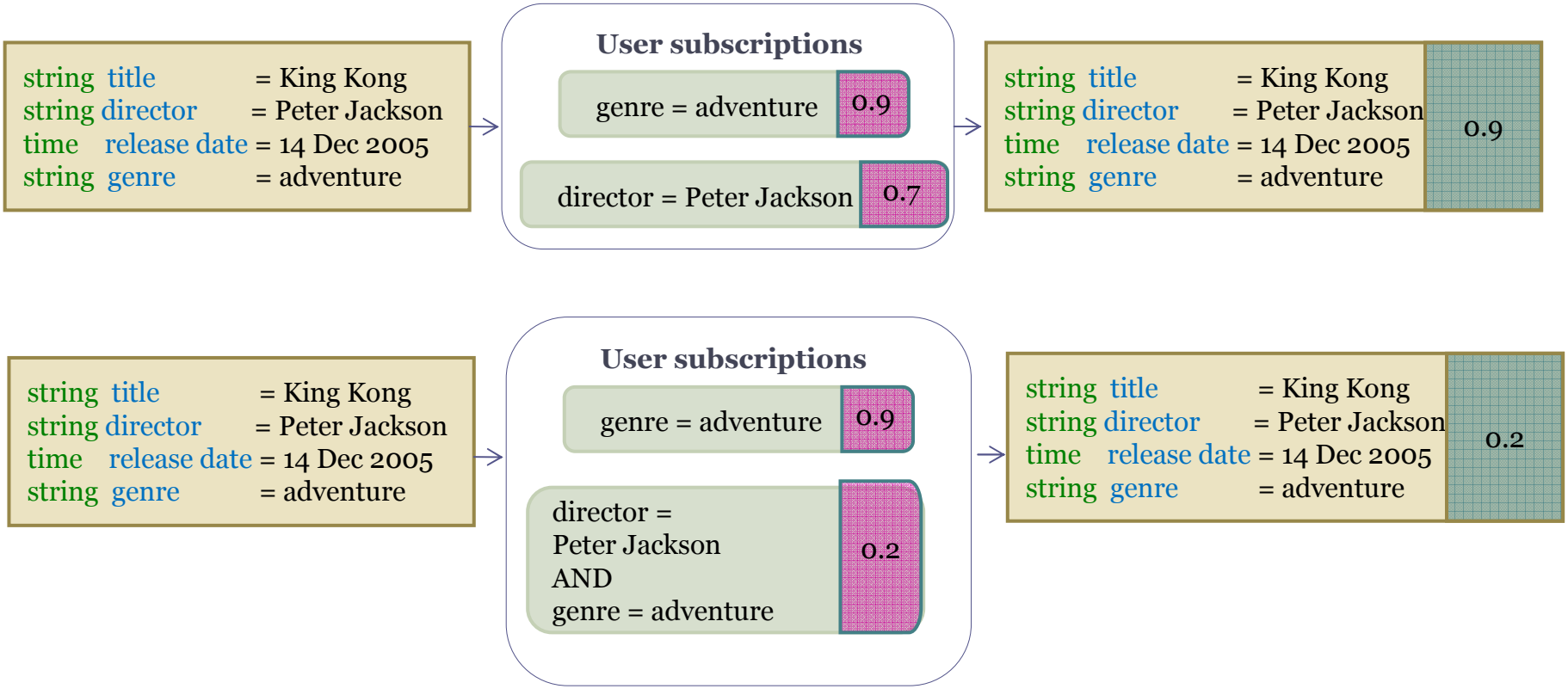
How?

- The event rank is computed based on the scores of the matching subscriptions

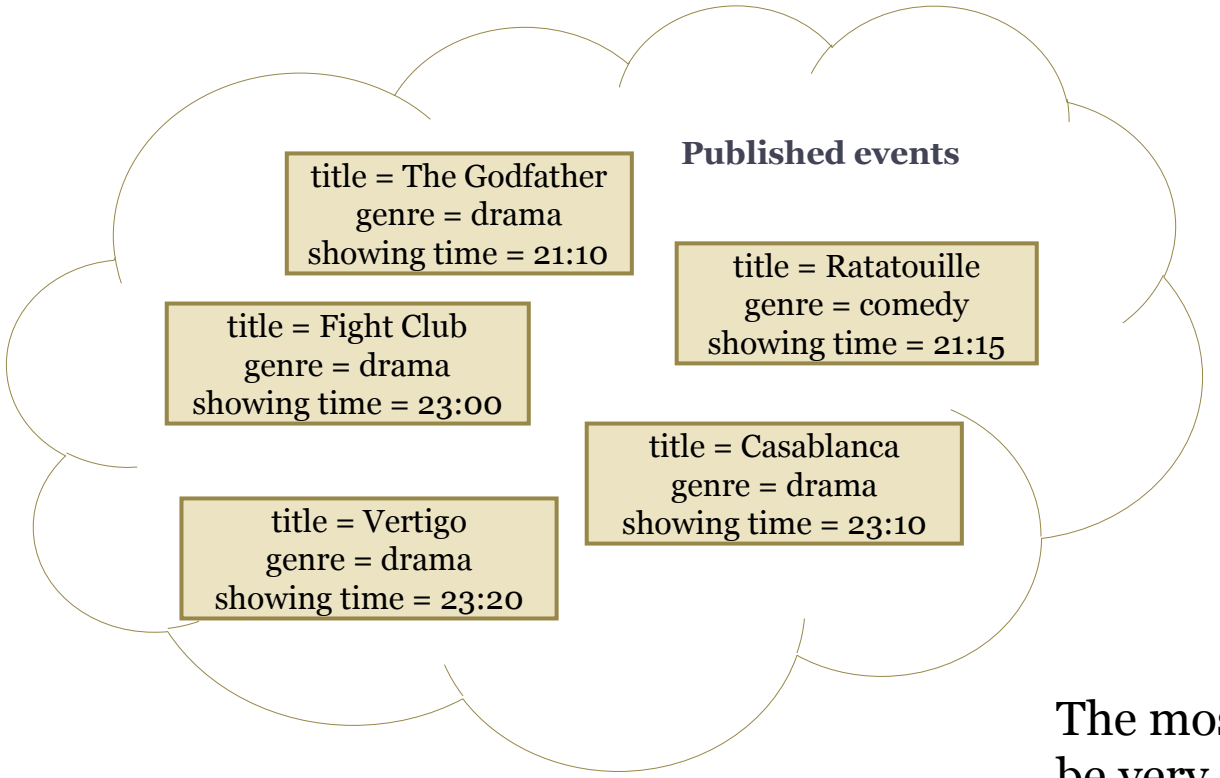
- In the case of more than one matching subscription:
  - Need to aggregate to produce a single rank
    - We use maximum and only the most specific matching

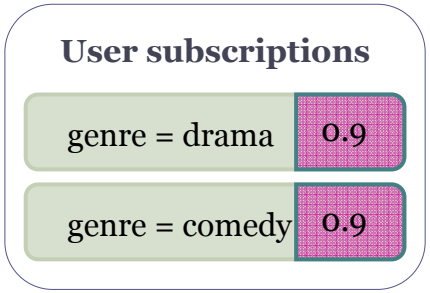
Preferential Pub/Sub

$\mathcal{F} = \max$



Spicing up data: Diversity





The most highly ranked events may be very similar to each other…

Diversity

We wish to retrieve results on a broader variety of user interests

Two different perspectives on achieving diversity:

- Avoid overlap: choose events that are dissimilar to each other
- Increase coverage: choose notifications that cover as many user interests as possible

How to measure diversity?

- Many alternative ways
- Common ground: measure similarity/distance among the selected items

Diverse top-k Delivery

Given a set M of n matching events, $|M| = n$, deliver a subset L, $L \subseteq M$, with cardinality k, such that,

$$div(L) = \max_{L' \subseteq M, |L'|=k} \{div(L')\}$$

This type of problem (k-dispersity in OS) is generally known to be NP-hard

- Note that the term "set" is used loosely. The order of events in L is important

Diversity Heuristic

We use a greedy heuristic to produce diverse subsets of events from a pool of candidate ones

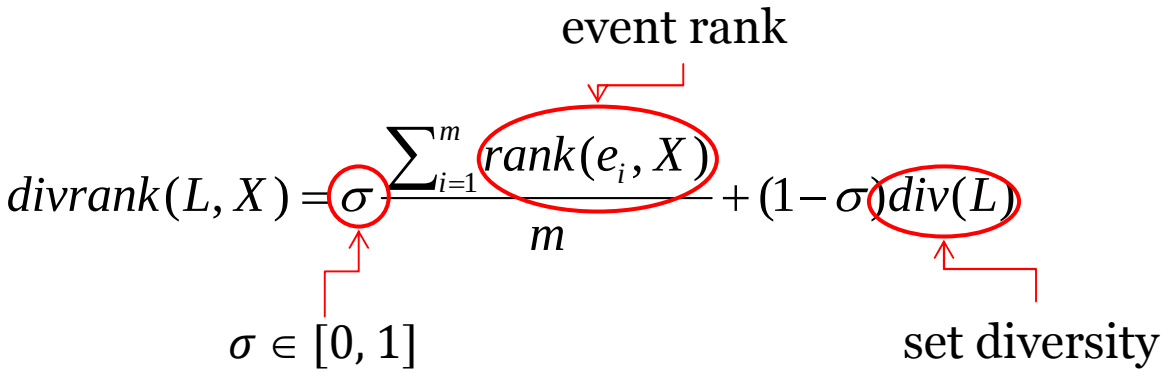
Basic idea:

- Select first the two furthest apart events from M and add them to L
- Select one-by-one the most diverse events:
  - Compute the distances $dis(e_i, L)$ for each $e_i \subseteq M \setminus L$
  - Add to L the event e_{add} with the maximum $dis(e_{add}, L)$
- Continue until k events have been selected

Combining Preferences with Diversity

We want to combine both relevance (as specified by user preferences) and diversity for our final ranking

Given a user X and a set of m events $L = \{e_1, \ldots, e_m\}$, the diversity-aware rank of L for X is:



Top-k preferred diversity-aware delivery

TOP-*K* PREFERRED DIVERSITY-AWARE DELIVERY

Given a set M of n matching events, $|M| = n$, for a user X, deliver a subset L, $L \subseteq M$, with cardinality k, such that, $divrank(L) = \max_{L' \subseteq M, |L'|=k} \{divrank(L')\}$.

- Modify the heuristic

Timing

New events are continuously published and matched

- Over which set of events of this stream do we apply preference ranking and diversification?

We consider 3 alternative delivery modes:

- Periodic delivery
- Sliding-window delivery
- History-based filtering

- Complexity (storage (buffer size) and time)
- Freshness
- Rank and diversity

PrefSIENA: our prototype

We have extended the SIENA [1,2] to include preferential subscriptions and delivery based on ranking and diversity for the three delivery modes

- http://www.cs.uoi.gr/~mdrosou/PrefSIENA

[1]A. Carzaniga, D.S. Rosenblum and A.L. Wolf. "*Design and evaluation of a wide-area event notification service*". ACM Trans. n Computer Syst., 19:332-383, 2001
[2] http://serl.cs.colorado.edu/~serl/dot/siena.html

Much more to do with Preferences

- Privacy with regards to preferences (for example, characterize the privacy of a specific profile)
- Diversity (metrics/algorithms)
- In terms of preferential publish/subscribe
  - Appropriate data structures (currently POSETS as in SIENA): some form of clustered overlays -> topic-based easier, use overlay networks?
  - Use coverage instead of diversity
  - Improve performance: Caching/Replication
  - Delivery Modes
  - Privacy
- …

Part 2: Overlay networks

Our Work on Overlays

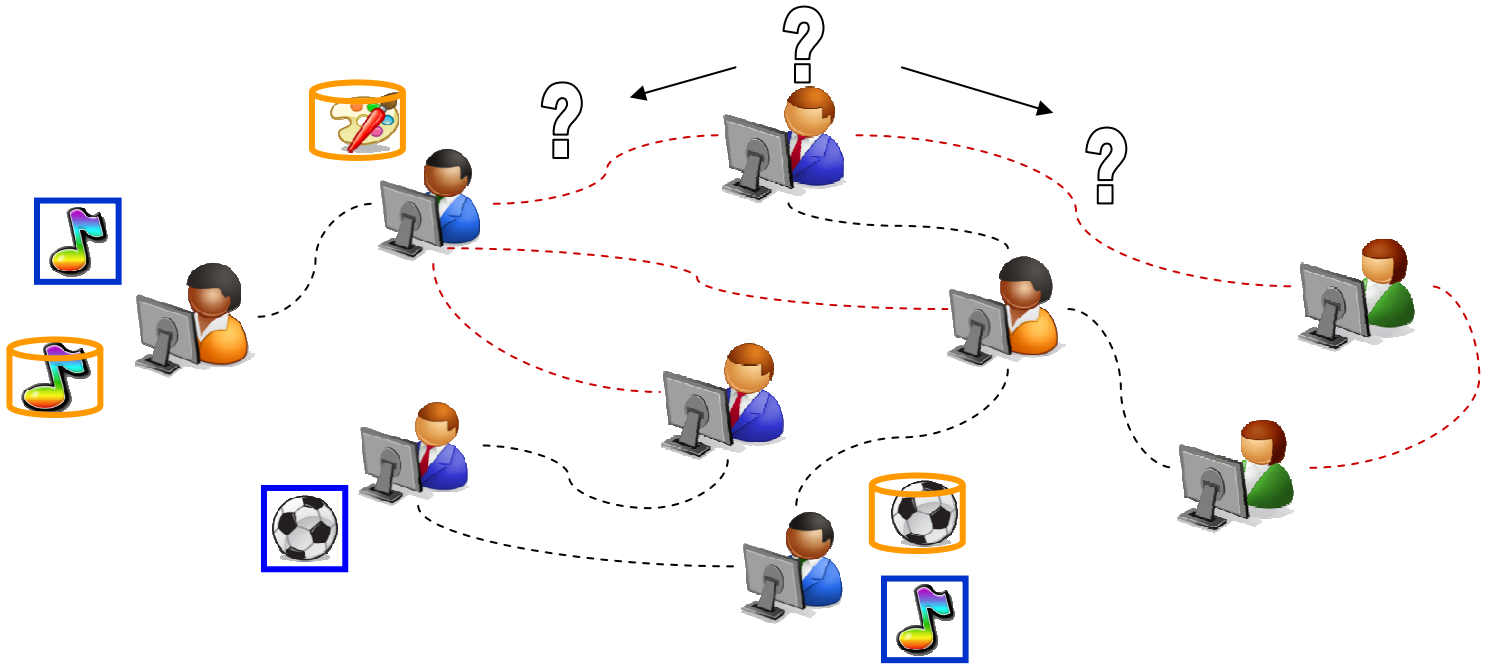
- Distributed XPath over Distributed XML Collections [**ICDE08**, **SIGMOD08**, **ICDE09**]
  - Cooperative caching for XML: use a DHT-based cache for indexing or sharing cache [SIGMOD08]
  - Clustering for relaxation [ICDE08, ICDE09]
- Cluster overlays as a game [**VLDB09**]

Joint work with

Georgia Koloniari **http://www.cs.uoi.gr/~kgeorgia**

and **Kostas Lillis**

Overlays for Sharing Distributed Content

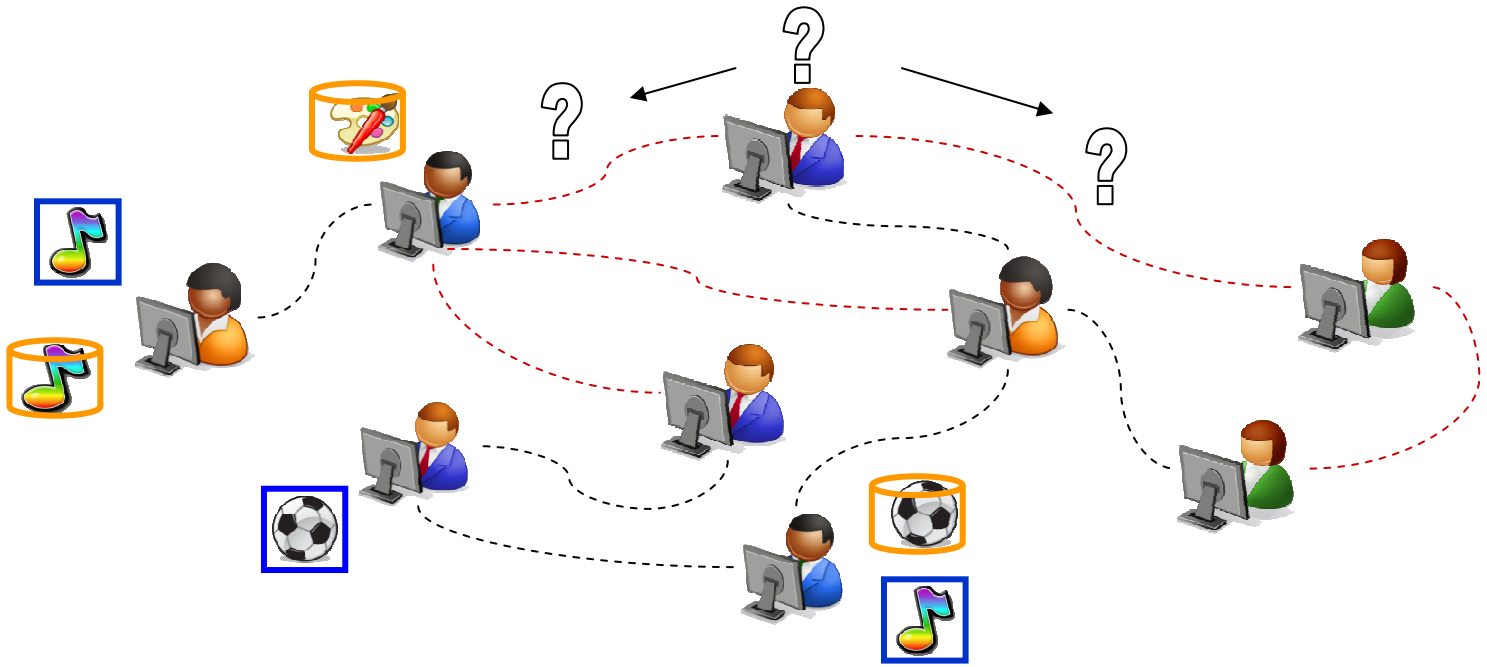


Dynamic Content Sharing Systems: P2P, social networks

Peers:

- Offer/Store content
- Request/Query for content

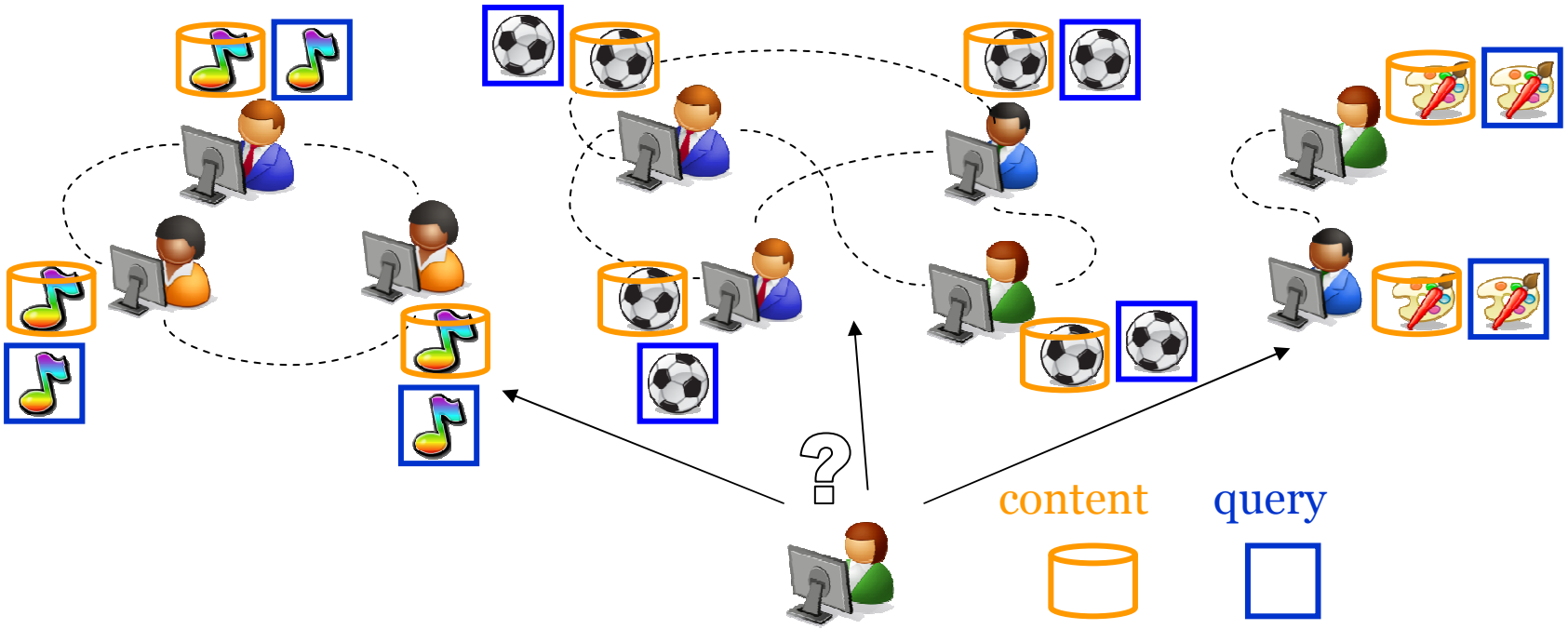
Overlay Networks



Peers connect with a subset of other peers

Overlay networks built on top of the physical networks

Clustered Overlays



Connect with “similar” peers

Why?

Once the relevant clusters for a query are identified, the peers in them maintain relevant content that can be exploited to evaluate and refine a query

Related Work on Cluster Overlays (among others)

- SONS [Stanf. Univ. '02]: clusters formed based on predefined classification hierarchies
- SETS [SIGIR '03]: peers partitioned in clusters corresponding to fixed globally known topic segments
- Cohen et al [INFOCOM '03]: based on a learning approach that generalizes and learns the semantic categories of the data
- Triantafillou et al [CIDR '03]: fixed set of clusters formed based on predefined semantic categories, focus on fair load distribution and reducing response times
- Garbacki et al [ICDCS '07]: superpeer-based architecture in which peers with common interests are organized based on their caches
- Doulkeridis et al [JSAC '07]: clustering applied first on the documents of each peer, and then on the feature vectors describing the derived clusters

Our Approach

Model the dynamics of distributed cluster creation

- Peers decide to join the clusters that maximize the recall of their queries
- Independent decisions

Use game theory to formulate the cluster formation problem

The Game

Let C be the set of possible clusters (can be dynamic)

- Each peer p_i is a player
- A player selects a strategy s_i which consists of the set of clusters the peer wants to join ($s_i \subseteq C$)
- The goal of the game for each peer is to select the strategy that minimizes a utility function
  - Utility function: maximize the recall of its own queries

Cluster Configuration: $S = \{s_1, s_2, .., s_{|P|}\}$

The Utility Function

Assumption: It costs less to evaluate the queries inside a clusters (that is, using the content provided by the peers inside the cluster) than to evaluate the queries outside the clusters

A peer wants to join the clusters – select the strategy - whose peers have the most results for its own queries

We quantify this through recall: the percentage of results of q when using only the content of peer q_i

$$r(q, p_i) = \frac{result(q, p_i)}{\sum_{p_k \in P} result(q, p_k)}$$

The Utility Function

How many results a peer p_i gets when it evaluates a query using only the peers in its clusters - in its strategy - or (equivalently) the results it loses, i.e., the results provided by clusters outside its strategy -> recall cost

membership cost

recall cost

$$pcost(p_i, S) = a * \sum_{c_k \in s_i} \frac{\theta(|c_k|)}{|P|} + \sum_{q \text{ in } Q(p_i)} \frac{num(q, Q(p_i))}{num(Q(p_i))} \sum_{p_j \notin P(s_i)} r(q, p_j)$$

All queries of p_i

$num(q, Q(p_i))$: number of appearances of q in the local workload $Q(p_i)$ of p_i

$num(Q(p_i))$: number of queries in the local workload $Q(p_i)$ of p_i

$P(s_i)$: peers that are members of the clusters in p_i's strategy

The Utility Function

Best strategy: join all clusters

Add a component to model the cost of cluster participation

What is this cost? Depends on:

- cluster size and
- cluster connectivity

The Utility Function

membership cost

recall cost

$$pcost(p_i, S) = a * \sum_{c_k \in s_i} \frac{\theta(|c_k|)}{|P|} + \sum_{qinQ(p_i)} \frac{num(q, Q(p_i))}{num(Q(p_i))} \sum_{p_j \notin P(s_i)} r(q, p_j)$$

The cost of cluster participation measured by a function θ

- monotonic with the size of the cluster $|c_i|$
- the topology within the cluster (linear, logarithmic, etc)

Tuning parameter a characterizes the relative cost between maintenance/update and query recall

Selfish vs Altruistic: improves the recall of other peers

$$pcontr(p_i, S) = \sum_{p_j \in P(s_i)} \sum_{q \in Q(p_j)} \frac{num(q, Q(p_j))}{num(Q(p_j))} r(q, p_i) - a * \sum_{c_k \in s_i} \frac{|c_k| \theta(|c_k|)}{|P|^2}$$

- The contribution to the queries of the other peers in the cluster
- The cost the other peers in a cluster pay when p_i joins

Hybrid Cost Function

$$hpcost(p_i, S) = dpcost(p_i, S) - (1-d) pcontr(p_i, S)$$

- $d \in [0,1]$ – degree of selfishness

Now, that we have defined the game, let us study its result …

Properties of the cluster configuration

- How to measure the overall quality (global cost) of the cluster configuration
- Does the game converge? Do we reach a stable configuration?
- How good (in terms of cost) is the stable configuration? (price of anarchy)

Global cost

Social Cost: the sum of the peers individual costs

$$SCost\ (S) = \sum_{p_i \in P} pcost(p_i, S)$$

Workload-based Cost: (from a query perspective) the average cost (recall loss) of all queries

$$WCost(S) = a \sum_{C_k \in C} \frac{|C_k|\,\theta(|C_k|)}{|P|} + \sum_{qinQ} \overbrace{\frac{num(q,Q)}{num(Q)}}^{\text{frequency of } q \text{ in global query workload}} \sum_{p_i, qinQ(p_i)} \frac{num(q,Q(p_i))}{num(Q(p(i))} \sum_{p_j \notin P(s_i)} r(q, p_j)$$

$\underbrace{}_{\text{frequency of } q \text{ in local query workload of } p_i}$

- Social cost treats all peers as equals, the workload-based cost favors more demanding peers
- If each peer gets an equal portion of the query workload, then proportional to each other

Global Contribution

In analogy to global cost:

Social contribution and workload contribution

- Social contribution favors queries popular locally to specific users, while workload contribution favors overall popular queries
- If the local distribution queries at each peer follows the global query distribution, then the two measures are proportional

If we ignore the membership cost, then the workload cost and workload contribution are complementary

Properties of the cluster configuration

- How to measure the overall quality (global cost) of the cluster configuration
- Does the game converge? Do we reach a stable configuration?
- How good (in terms of cost) is the stable configuration? (price of anarchy)

Stability

Nash equilibrium: No peer has an incentive to change its strategy

$$pcost(p_i, S) \leq pcost(p_i, S'), \forall S' = S - \{s_i\} \cup \{s_i'\}$$

Lemma:

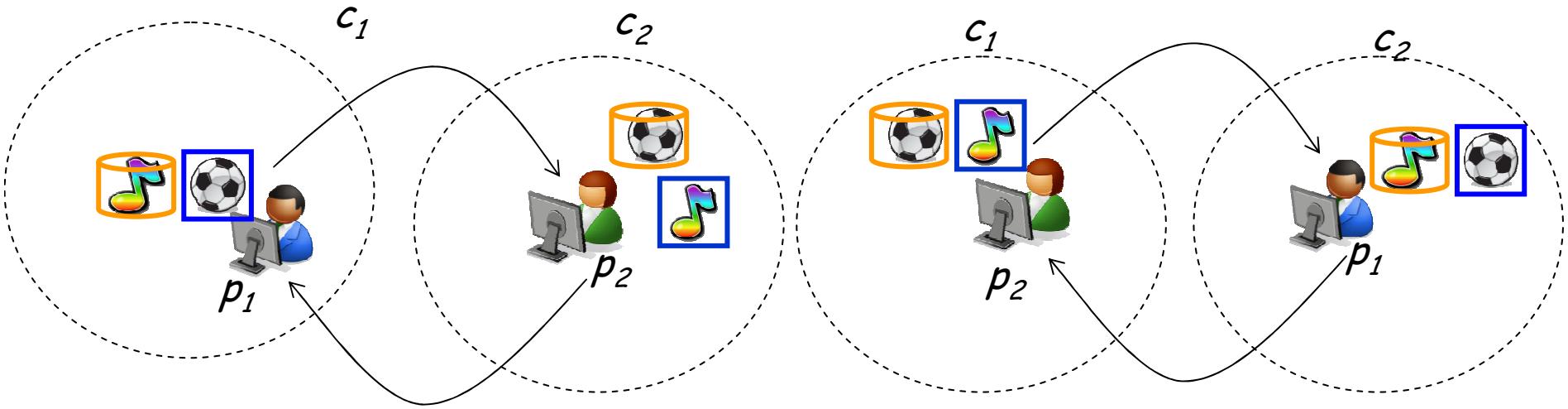
In any stable state, there are no clusters c_i, c_j such that $c_i \subseteq c_j$, $i \neq j$

Corollary:

When a peer forms a cluster by itself, it cannot belong to any other cluster

Stability

In the general case, the cluster formation game is not stable



Payoff Table

	p_2 joins c_1	p_2 joins c_2
p_1 joins c_1	α, α	$\alpha/2+1, \alpha/2$
p_1 joins c_2	$\alpha/2+1, \alpha/2$	α, α

Properties of the cluster configuration

- How to measure the overall quality (global cost) of the cluster configuration
- Does the game converge? Do we reach a stable configuration?
- How good (in terms of cost) is the stable configuration? (price of anarchy)

Optimality and the Price of Anarchy

The social optimum is obtained by minimizing the social cost measure over all possible configurations

Price of Anarchy: ratio between the social cost of the worst Nash equilibrium to the social optimum

In general, stable states are not necessarily optimal

Case Studies

We consider a number of scenarios (cases) and study (theoretically) whether/when specific cluster configurations are stable and the price of anarchy

Various Cases, among others:

Case I: No underlying clustering

Case II: “Perfect” symmetry

Cluster Configurations:

A: all peers in single cluster

B: all peers a cluster by their selves

C: *k* clusters

Case Studies

Conditions for stability

	Case (I.A)	Case (I.B)	Case (I.C)
Cost-based	$\alpha \le \frac{\mid N \mid -1}{\theta(\mid N \mid) - \theta(1)}$	$\alpha \ge \frac{k}{k\theta(2) - \theta(1)}$	$\alpha \le \frac{\mid c \mid -1}{\theta(\mid c \mid) - \theta(1)}, \alpha \ge \frac{\mid c \mid}{\theta(\mid c \mid +1)}, a \ge \frac{k(\mid c \mid -1)+1}{k\theta(\mid c \mid +1) - \theta(\mid c \mid)}$
Linear θ	$\alpha \le \frac{1}{\varphi}$	$\alpha \ge \frac{k}{(2k-1)\varphi}$	$\alpha \le \frac{1}{\varphi}, \alpha \ge \frac{\mid c \mid}{\varphi(\mid c \mid +1)}, a \ge \frac{k(\mid c \mid -1)+1}{k\varphi(\mid c \mid +1) - \varphi \mid c \mid}$
	Case (II.A)	**Case (II.B)**	**Case (II.C)**
Cost-based	$\alpha \le \frac{\mid N \mid (\mid N \mid -m)}{m\theta(\mid N \mid) - \mid N \mid \theta(1)}$	$\alpha \ge \frac{m}{\theta(2) - \theta(1)}$	$\frac{\mid N \mid^2 -m}{\mid N \mid (\theta(\mid N \mid / m +1) - \theta(\mid N \mid / m)} \le \alpha \le \frac{(\mid N \mid -m)}{\theta(\mid N \mid / m) - \theta(1)}$
Linear θ	$\alpha \le \frac{\mid N \mid -m}{\varphi(m-1)}$	$\alpha \ge \frac{m}{\varphi}$	$\alpha \ge \frac{\mid N \mid^2 -m}{\varphi \mid N \mid}, \alpha \ge \frac{(\mid N \mid -m)}{\varphi(\mid N \mid / m -1)}$

Linear θ: $\theta(n) = \varphi n$

To Recap:

Model cluster formation/maintenance as a strategy game

Study:

- social cost
- stability
- price of anarchy

Next:

How to play the game? (We need a (practical) protocol to implement it)

Playing the Game

For simplicity, no overlap among clusters: each peer p_i belongs to a single cluster ($s_i = \{c_l\}$), $c_l \in C_{cur}$ (set of non empty clusters)

Given an instance of the system (S_{cur}), when it is its turn to play, p_i considers all possible configurations S_j that differ with S_{cur} only in s_i

For non overlapping clusters, p_i has two options:

- Move to a different existing cluster c_v
- If $|c_l| \neq 1$, form a cluster by its own

Selfish, altruistic and hybrid behavior

Playing the Game

Each peer:

- evaluates its cost or contribution for each alternative
- selects the best strategy
  - computes the gain for the best strategy (cluster)
  - if $gain > 0$, p_i moves to best cluster

$$gain_{p_i} = pcost(p_i, S_{cur}) - pcost(p_i, S_{new})$$

Estimation:

Annotate each result with the cluster it came from (for instance, cluster id = first cluster in)

The protocol is uncoordinated and based on local decisions made by each peer independently

Note:

Playing the game from an initial configuration where all peers belong to a single cluster or form a cluster by their own corresponds to cluster creation

When to play:

- Event-based: after it becomes aware of a relevant event
  - Selfish peer: one of their queries was evaluated
  - Altruistic peer: provided results to a query

  batch-based variation (after a batch of relevant events)

- Trigger-based: when it registers a change in its gain
  - requires continuously monitoring the gain

Controlling the overhead:

- Stopping Condition: A peer moves only if its gain is greater than a predefined threshold ε (ε-stability)

- Playing Probability: A peer does not play at each of its turns, but with a probability *Pr*

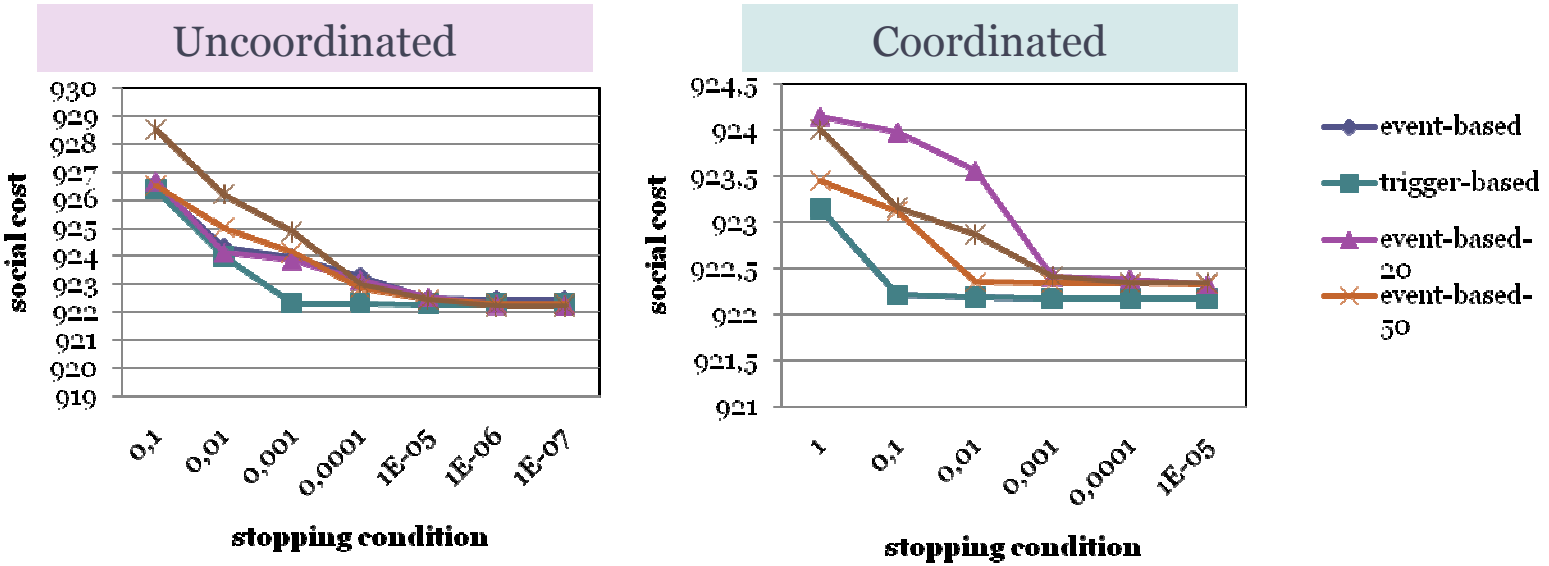
- Quota: Each peer is assigned *n* moves for a time period T_q

Coordinated Protocol

Coordinated protocol:

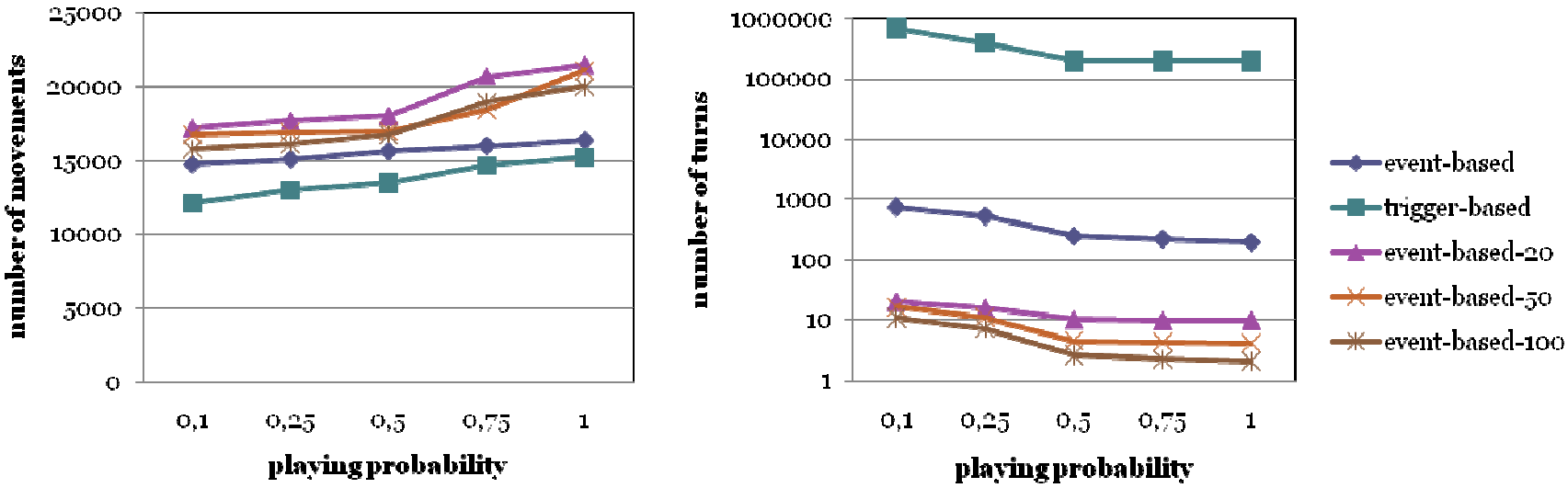
- cluster representatives gather and exchange the relocation requests from their clusters
- requests are sorted according to gain and granted in that order

Uncoordinated Vs Coordinated



- The uncoordinated protocol performs as well as the coordinated one without the additional coordination overhead
- The trigger-based protocol is the most expensive variation and the batch-based the less expensive one --- The trigger-based variation adjusts faster to changes

Tuning Parameters



- The stopping condition is the main factor determining the value of the achieved social cost
- The playing probability and quota reduce the number of moves but increase the number of turns

Cluster Formation

Setting:

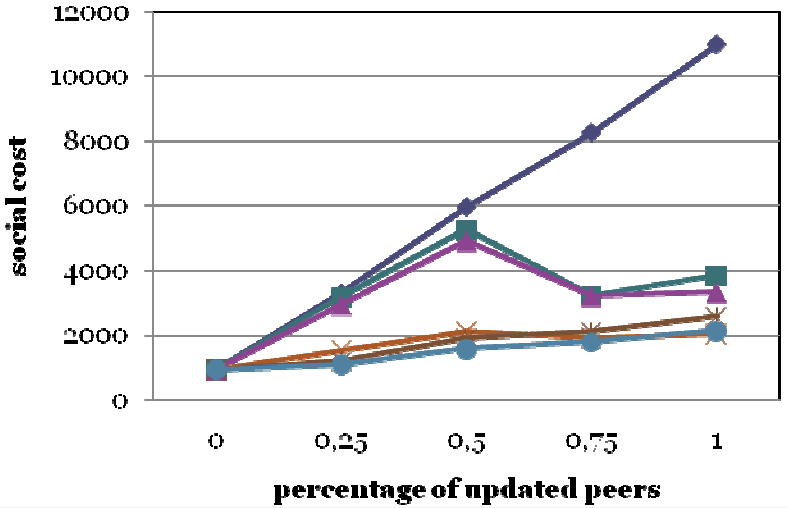
- Starting from different initial configurations (all peers in a single cluster, each peer a cluster by its own, x random clusters, etc)
- Using symmetric, asymmetric, and uniform peers
- With selfish, altruistic and mixed peer populations

Results:

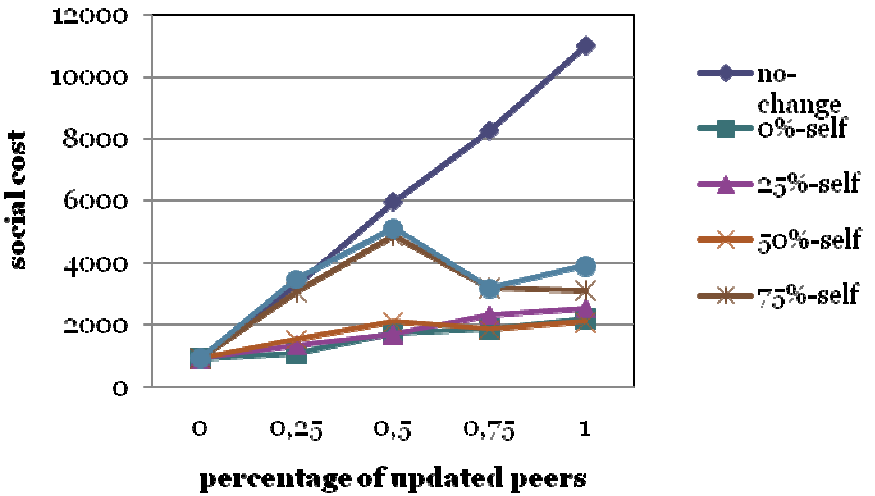
- The reformulation protocol identifies the underlying clusters if they exist
- It does not require a priori definition of the target number of clusters
- Its social cost in some cases (i.e. for symmetric and uniform peers) reaches a value close to the social optimum

Cluster Adaptation

Workload Updates



Content Updates



- For different update scenarios our protocol copes with changes efficiently
- Selfish peers react more efficiently to workload changes, while altruistic to content changes
- Reclustering reduces social cost by 10%, but requires 250 turns while the reformulation protocol only 10

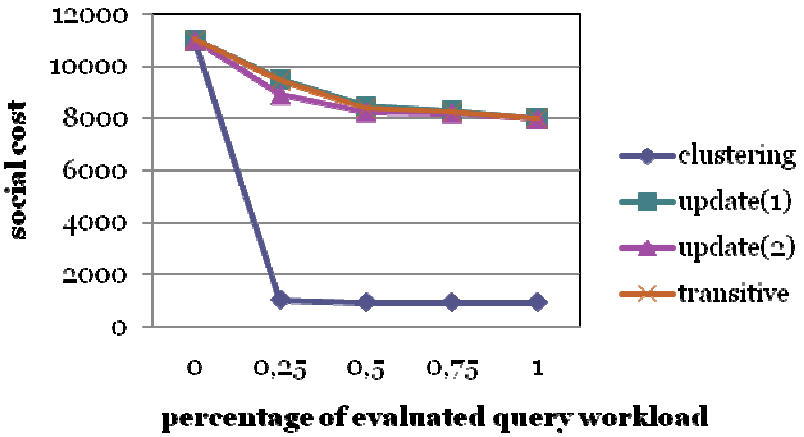
Clustering vs Caching

Consider a cache scheme:

- Each peer caches the peers that provided results to previous queries
- Future queries are forwarded to peers in the cache first

- Transitive variation: If peers that receive a query also use the peers in their own cache
- Recall-based cache replacement: Peers in the cache are sorted based on recall – Replace the one that provides the smaller recall
- Dynamic cache update: Cache is updated after each query

Clustering vs Caching



- Cluster works better for symmetric peers
- For asymmetric peers, using an efficient cluster topology achieves results similar to caching
- Clustering adapts to changes faster (replaces all links at once)

Issues on Overlays to be Explored Further

- Modeling aspects (concave utility function that may lead to convergence, other forms of altruism, modeling cost for cluster discovery, etc)
- Practical protocols for allowing overlapping clusters, further experiments ..
- Apply/adjust the cluster formation game for friends discovery in social networks
- Compare with traditional goals in clustering applications (minimizing the distance between members of a cluster - maximizing the distance among members of different clusters)
- Use different criteria (besides recall) such as diversity, for determining the quality in clustered overlays

Thank you!

Room **KACB 2228**

Until **September 18**