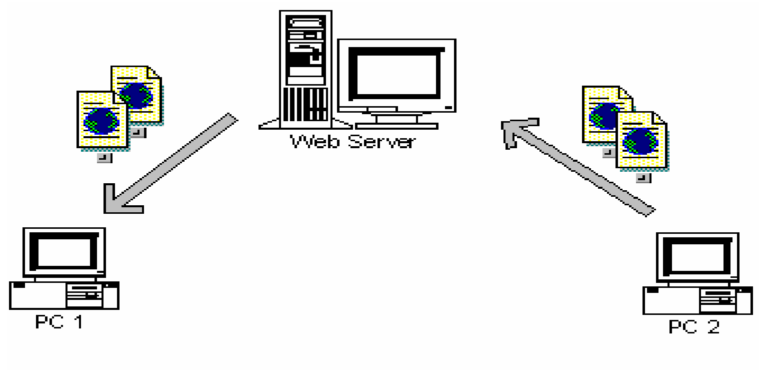
Data Management in P2P Systems: Challenges and Research Issues

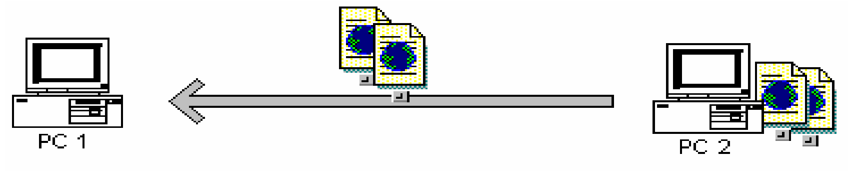
Timos Sellis

National Technical Univ. of Athens

What is P2P?



Client Server Architecture



Peer-to-Peer Architecture

P2P: an old concept

- P2P is not a new technology
  - *Oct. 29 1969: first transmission UCLA → Stanford Research Inst. (SRI) [UCSB, U. Of Utah]; Peer computing status among independent computing sites*
- 1970s-1980s: ARPANET completed (1978); Usenet and DNS
- 1990s: NSFnet and Internet explosion
- 1996: ICQ (bypassing DNS by offering own addressing scheme and allowing end points to be directly addressable with each other)

NAPSTER boosts P2P technology

- 1999: Napster is born
  - universities notice first
  - mid-2001: shut down service; other apps emerge since 2000
- 2002: Napster is gone…
- April 30, 2002:

  A study from WebSense indicates that the number of file-swapping, and peer-to-peer websites, has grown by 535 percent in the past year, despite legal efforts to have them shut down.

  According to the study findings, the number of p2p websites totals nearly 38,000.

The Evolution of P2P systems

- First generation – centralized P2P systems
  - *E.g. Napster, SETI@home*
- Second generation –decentralized & unstructured P2P systems
  - *E.g. Gnutella*
- Third generation—structured P2P systems
  - *DHT systems (CAN/Chord/Pastry/Tapestry)*
  - *Skip-list based systems*
- ….

P2P Perspective: Self-Organized Systems

P2P Paradigm is characterized by:

- no centralized control
- adaptable behavior
- self-organization
  - *distribution of control ⇒ decentralization*
  - *local interactions, information and decision ⇒ autonomy*
- emergence of global structures
- failure resilience and scalability

P2P Paradigm Dimensions

- Structure – Framework
  - *Servers and clients*
- Organization
  - *Structured vs. unstructured overlays*
- Managed and stored data
  - *Files vs. (semi)-structured data*
- Functionality-Applications
  - *Sharing, resource management, web services, collaboration etc*

Structure- Frameworks

Main differences of different structures:
how information is shared and how much information is shared

- Models:
  - *Centralized*
  - *Decentralized*
  - *Controlled-decentralized | Hybrids*
- Components:

  - client, server, serv*e*nts

Centralized: pros and cons

Centralized servers keep a catalog of available files ⇒ Napster

Pros

- More effective, comprehensive searches
- Access is controlled

Cons

- System has single points of entry; one fail could bring the whole system down
- Broken links, out of date information.

Decentralized Framework

- Every user acts as a client, a server or both (serv*e*nt).
- User connects to framework and becomes a member of the community, allowing others to connect through him/her
- Users “talk” directly to other users with no intermediate or central authority
- No one entity controls the information that passes through the community

  ⇒Gnutella, Freenet

Decentralized: Pros and Cons

Pros

- Peers “talk” directly with no central authority. Nobody owns the Gnutella Network and nobody can shut it down.
- Isolated node failure can quickly and automatically be worked around.
- Free loading
- Scalability

Cons

- Searches are less effective and can be slow.

Hybrids: Controlled decentralization

- Characteristics of both centralized and decentralized frameworks, i.e. systems with ***super-peers***
- User's computer may act as **client**, a **server** or a **servent**; there are server operators which may control which clients and/or servents are allowed to access a particular server.
- Examples: Morpheus, Groove

Organization: Overlay Networks

- Structured vs. unstructured overlays
- Structured:
  - data is indexed and stored at peers with pre-specified characteristics, e.g. peer id
  - Peers are connected to specific peers
- Unstructured:
  - No pre-specified connection of peers and storage/indexing of data

Managed and Stored Data

- Simple files:
  - search is performed with keywords for file names and file extensions
- Structured or semi-structured data ⇒ *P2P Databases*:
  - XML data
  - Relational data
  - RDF data

Key Application/Function Areas

- File sharing / Content distribution:
  - *Not just mp3s/media sharing*
  - *Distributed searching: Used to easily lookup and share files and offer content management*
- Instant messaging:
  - *Exchanging messages and chatting ⇒ Jabber, IM*
- Distributed computation:
  - *Use under utilized Internet and/or network resources for improving computation and data analysis ⇒* Grid Computing
- P2P groupware / Collaboration:
  - *Cooperative publishing, messaging, group project management. Secure environments are offered in some products*

What Has to Be Done

Issues to be worked out:

- Reliability

  Is the provided content or the provider itself reliable? How can we assure these?

- Resource management

  How can we achieve efficient collaboration of autonomous resources?

- Security & Privacy

  How can we guarantee security together with privacy without rescinding peer autonomy?

- Intellectual Rights

  Legal notions of copyright and intellectual property need to adapt to the sharing model of the P2P paradigm

Research Areas

- Peer discovery and group management
- Data location and placement
- Reliable and efficient file exchange
- Security/privacy/anonymity/trust

Relational data sharing in Unstructured P2P vs. Distributed DB

P2P	Distributed Database Systems
Nodes can join and leave the network anytime.	Nodes are added/removed from the network in a controlled manner.
Usually no predetermined (global) schema among nodes. Queries: Keywords	Have some knowledge of a shared schema. Queries: SQL
Answers to queries are typically incomplete (by "completeness" we mean all answers that satisfy a query)	Can actually retrieve the complete set of answers.
Content location is typically by "word-of-mouth" e.g., node routes query to its neighbors and so on…	Exact location to direct the query is typically known.

P2P & DB Systems

	P2P	DB
Flexibility	✓	✗
Decentralized	✓	✗
Strong Semantics	✗	✓
Powerful query facilities	✗	✓
Fault Tolerance	✓	✗
Lightweight	✓	✗
Transactions & Concurrency Control	✗	✓

P2P + DB = ?

- P2P Database? No!
  - *ACID transactional guarantees do not scale, nor does the everyday user want ACID semantics*
  - *Much too heavyweight of a solution for the everyday user*
- Query Processing on P2P!
  - *Both P2P and DBs do data location and movement*
  - *Can be naturally unified (lessons in both directions)*
  - *P2P brings scalability & flexibility*
    DB brings relational model & query facilities

Taken from Hellerstein's group ppt

Many New Challenges

- Relative to other parallel/distributed systems
  - *Partial failure*
  - *Churn*
  - *Few guarantees on transport, storage, etc.*
  - *Huge optimization space*
  - *Network bottlenecks & other resource constraints*
  - *No administrative organizations*
  - *Trust issues: security, privacy, incentives*
- Relative to IP networking
  - *Much higher function, more flexible*
  - *Much less controllable/predictable*

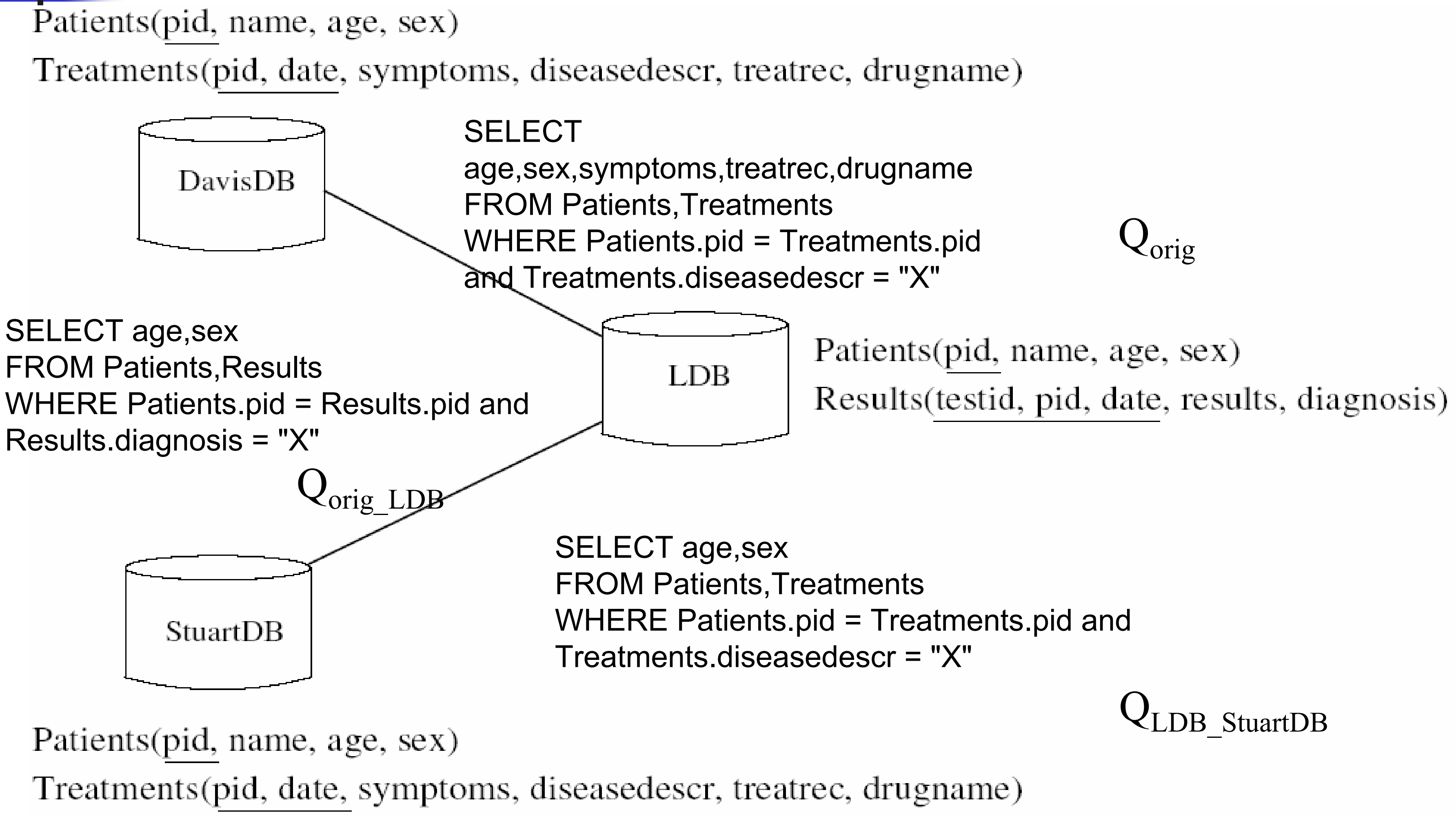
Data management in P2P systems

- Current research focuses on
  - *Decentralized schema mappings*
    - PeerDB: unstruct. network, keyword search only
  - *Extending DHT for complex querying*
    - PIER : exact-match and join queries
  - ***Query reformulation***
    - Edutella: super-peer, RDF-based schemas
    - Piazza: graph of pair-wise schema mappings
  - *Replication*
    - generally limited to static read-only files
    - P-Grid addresses updates in structured networks

Our model of P2P databases

- All peers hold relational databases
- There is no mediation or “global schema”
- Peers only have a few **acquaintees** (neighbors), i.e. other peers they “know” and can “talk to”
- Peers receive queries, answer them and at the same time **propagate** them to their neighbors
- Peers use mappings to map their own schema to their neighbor’s schemas in order to correctly translate a query and propagate it (Query rewriting)
- There is ad-hoc and intermittent connectivity

Query Answering: Motivating Application



Patients(pid, name, age, sex)

Treatments(pid, date, symptoms, diseasedescr, treatrec, drugname)

```
SELECT
age,sex,symptoms,treatrec,drugname
FROM Patients,Treatments
WHERE Patients.pid = Treatments.pid
and Treatments.diseasedescr = "X"
```

Q_{orig}

```
SELECT age,sex
FROM Patients,Results
WHERE Patients.pid = Results.pid and
Results.diagnosis = "X"
```

Q_{orig_LDB}

Patients(pid, name, age, sex)

Results(testid, pid, date, results, diagnosis)

```
SELECT age,sex
FROM Patients,Treatments
WHERE Patients.pid = Treatments.pid and
Treatments.diseasedescr = "X"
```

$Q_{LDB\_StuartDB}$

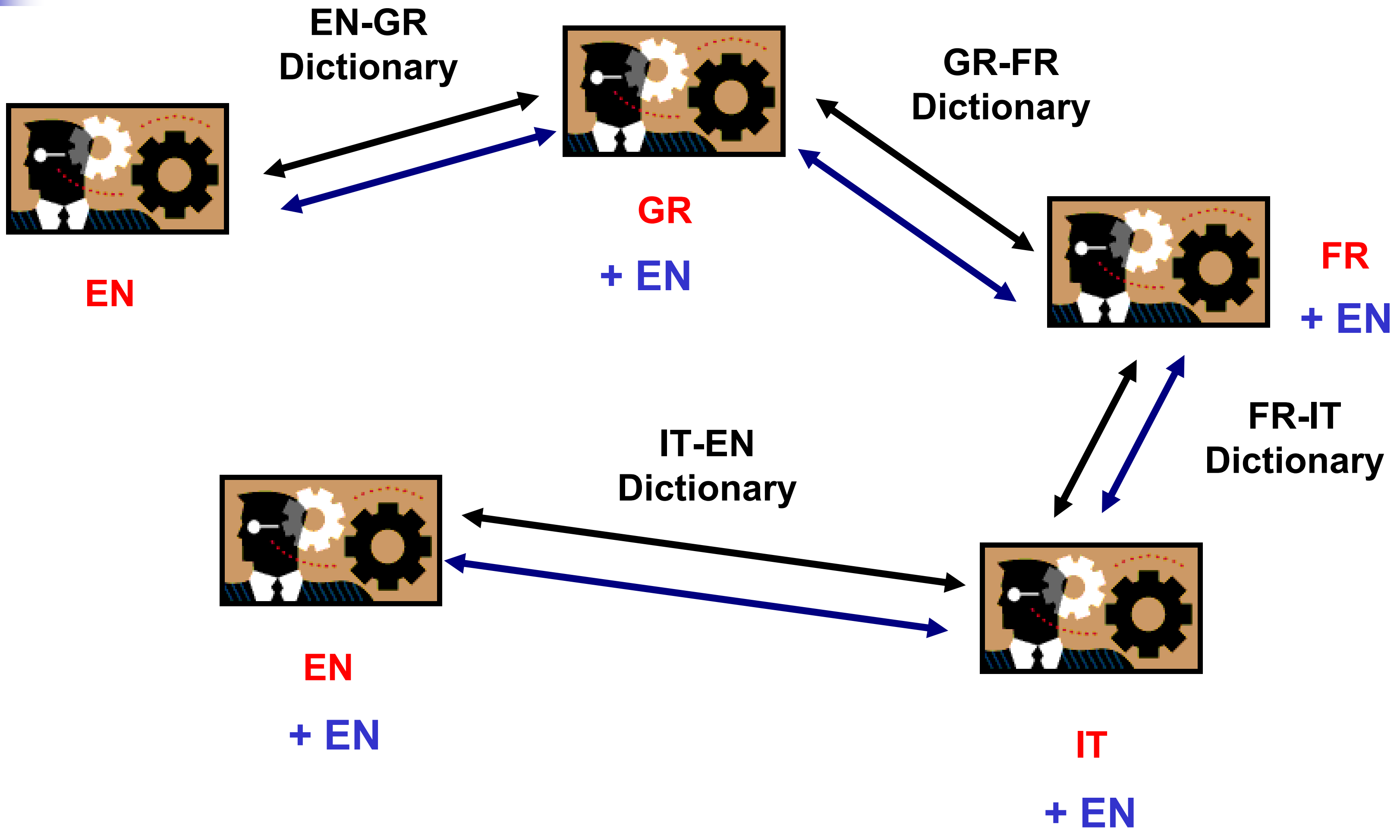
Patients(pid, name, age, sex)

Treatments(pid, date, symptoms, diseasedescr, treatrec, drugname)

Traditional Query Propagation (Contd.)

- Each peer that receives a query Q
  - *Rewrites it according to its local schema (Q')*
  - *Answers Q'*
  - *Forwards Q' to one or more of its neighbors*
- Peers with dissimilar schemas hide nodes rich in information
- Bad rewritings have the same effect
- Local-only knowledge in the overlay puts constraints on the propagation of a query
  - *Bandwidth limitations*
  - *Cannot keep track of all nodes!*

People have a similar problem



Overview of Our Framework - GrouPeer

- Forward **both** the **original** and the **rewritten** version of Q
- Answer the one most similar to the local schema
  - *Similarity function*

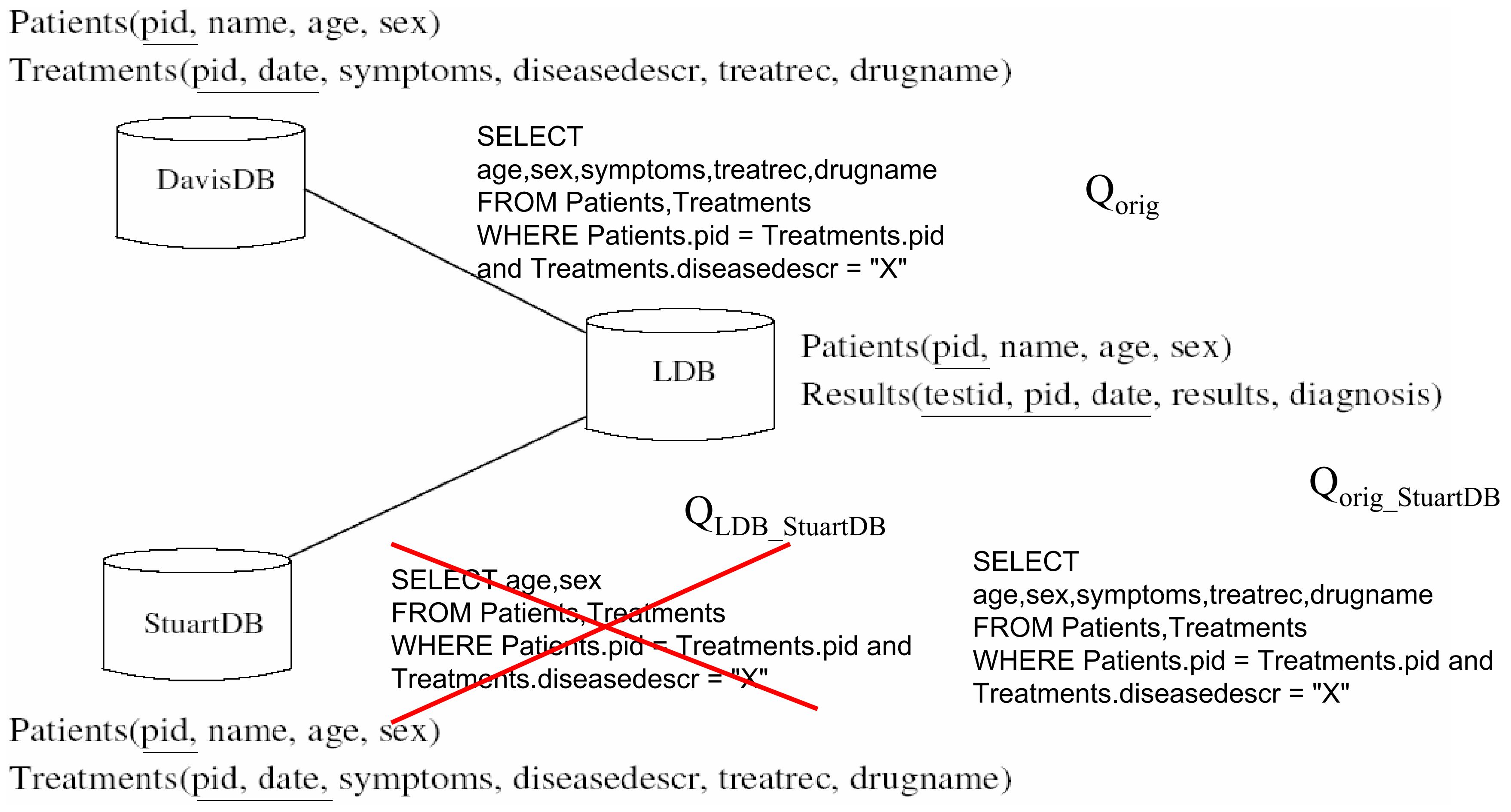
at the same time …..

- Overlay restructuring according to evaluation of the results (learning new neighbors)

but do this in an…..

- Intelligent, bandwidth-efficient query forwarding with informed walks

Answering the Original Query



Patients(pid, name, age, sex)
Treatments(pid, date, symptoms, diseasedescr, treatrec, drugname)
DavisDB
LDB
StuartDB
Patients(pid, name, age, sex)
Results(testid, pid, date, results, diagnosis)
Patients(pid, name, age, sex)
Treatments(pid, date, symptoms, diseasedescr, treatrec, drugname)

Q_{orig}

```
SELECT
age,sex,symptoms,treatrec,drugname
FROM Patients,Treatments
WHERE Patients.pid = Treatments.pid
and Treatments.diseasedescr = "X"
```

$Q_{LDB\_StuartDB}$

~~SELECT age,sex~~
~~FROM Patients,Treatments~~
~~WHERE Patients.pid = Treatments.pid and~~
~~Treatments.diseasedescr = "X"~~

$Q_{orig\_StuartDB}$

```
SELECT
age,sex,symptoms,treatrec,drugname
FROM Patients,Treatments
WHERE Patients.pid = Treatments.pid and
Treatments.diseasedescr = "X"
```

Similarity Measure

- Simple structural similarity between 2 versions of Q: $M_{sim}(Q_{v1}, Q_{v2})$ correlates simple conceptual similarities of structural query elements (i.e. 'select' attributes and 'where' conditions)

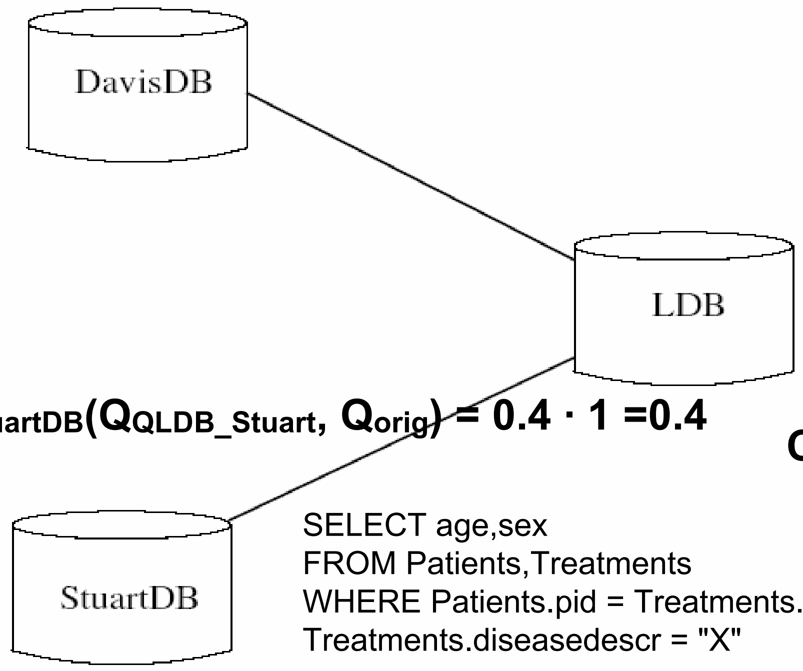
- Different at each peer
  - *Schema matching methodology*
  - *Individual preferences*

Confidence Parameter θ

- Express each peer's individuality
  - *Schema matching ability*
  - *Query structure*
  - $\theta = \theta_p(Q)$
- Each peer decides which query to answer (original, rewritten) according to:

$$Cov_p(Q_2, Q_1) = M_{sim\ p}(Q_1, Q_2) \cdot \theta_p(Q_1)$$

Example Query Answering



Patients(pid, name, age, sex)
Treatments(pid, date, symptoms, diseasedescr, treatrec, drugname)

Patients(pid, name, age, sex)
Results(testid, pid, date, results, diagnosis)

$Cov_{StuartDB}(Q_{QLDB\_Stuart}, Q_{orig}) = 0.4 \cdot 1 = 0.4$

$Cov_{StuartDB}(Q_{Qorig\_Stuart}, Q_{orig}) = 1.0 \cdot 0.5 = 0.5$

SELECT age,sex
FROM Patients,Treatments
WHERE Patients.pid = Treatments.pid and
Treatments.diseasedescr = "X"

SELECT
age,sex,symptoms,treatrec,drugname
FROM Patients,Treatments
WHERE Patients.pid = Treatments.pid and
Treatments.diseasedescr = "X"

Patients(pid, name, age, sex)
Treatments(pid, date, symptoms, diseasedescr, treatrec, drugname)

Learning Process

- Data providers return to the query originator:
  - **Which** version they answered
  - **Successively rewritten** query
  - **Locally rewritten** query
  - **Answer** tuples
- The originators evaluate the answers:

$$Ev(Ans) = Cov(Q_x, Q) \cdot max(1, w_Q \cdot |Res(Q_x)|)$$

$$Cov(Q_x, Q) = M_{sim}(Q, Q_x) \cdot \theta_{local}(Q)$$

Semantic Grouping of Peers

- The learning process leads to semantic clustering of the P2P network
- Each semantic cluster can create an explicit interest group:
  - *Group Initiator: Peer that initiates the group creation*
  - *Group Schema: An abstract schema with most common schema elements of the cluster*

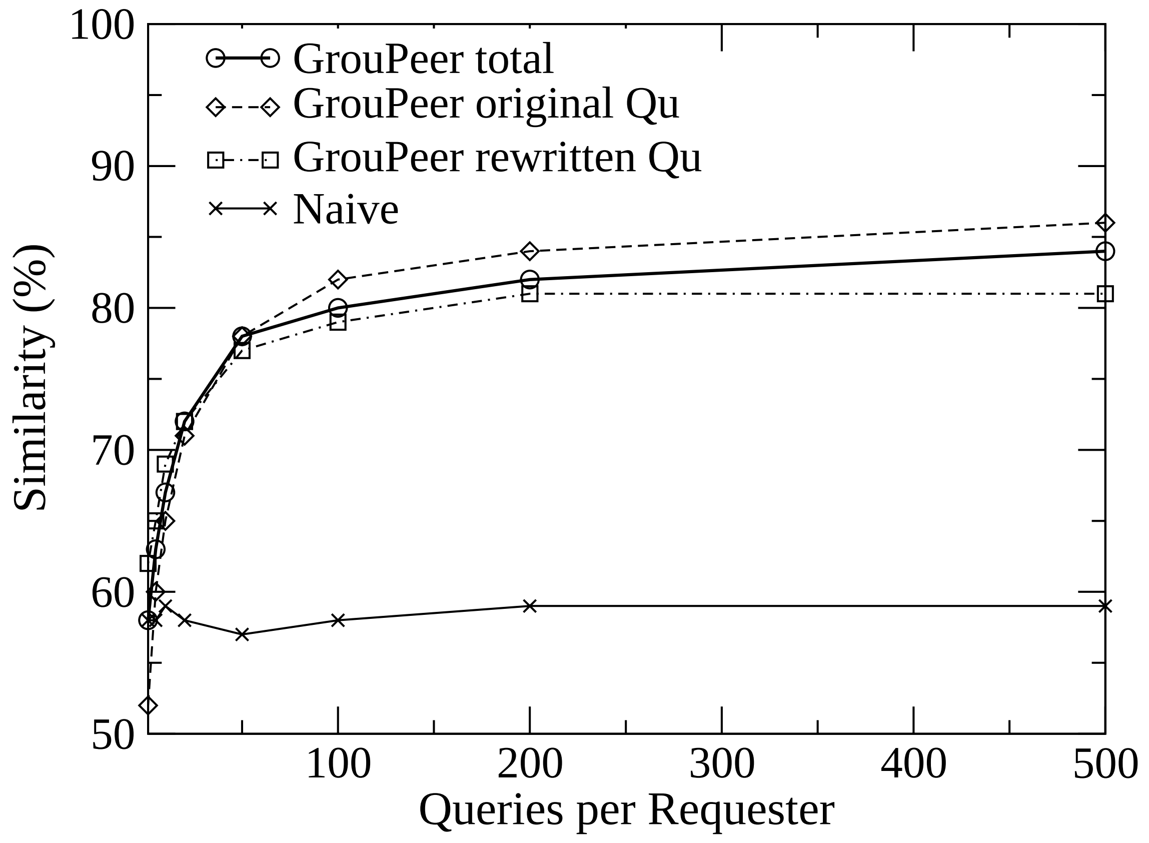
Routing and Restructuring in the Overlay

- Informed walks instead of blind forwarding
  - *K walkers, TTL steps each*
  - *Choose neighbors according to schema similarity*
- Adding an acquaintance p:
  - *When* $\theta_p = \theta_{local}$
  - *More than T replies from p*
  - *Impose an maximum number of neighbors*
- Dropping an acquaintance
  - *Low(est) similarity neighbors*
  - *Only if they don't get disconnected*

Simulation Methodology

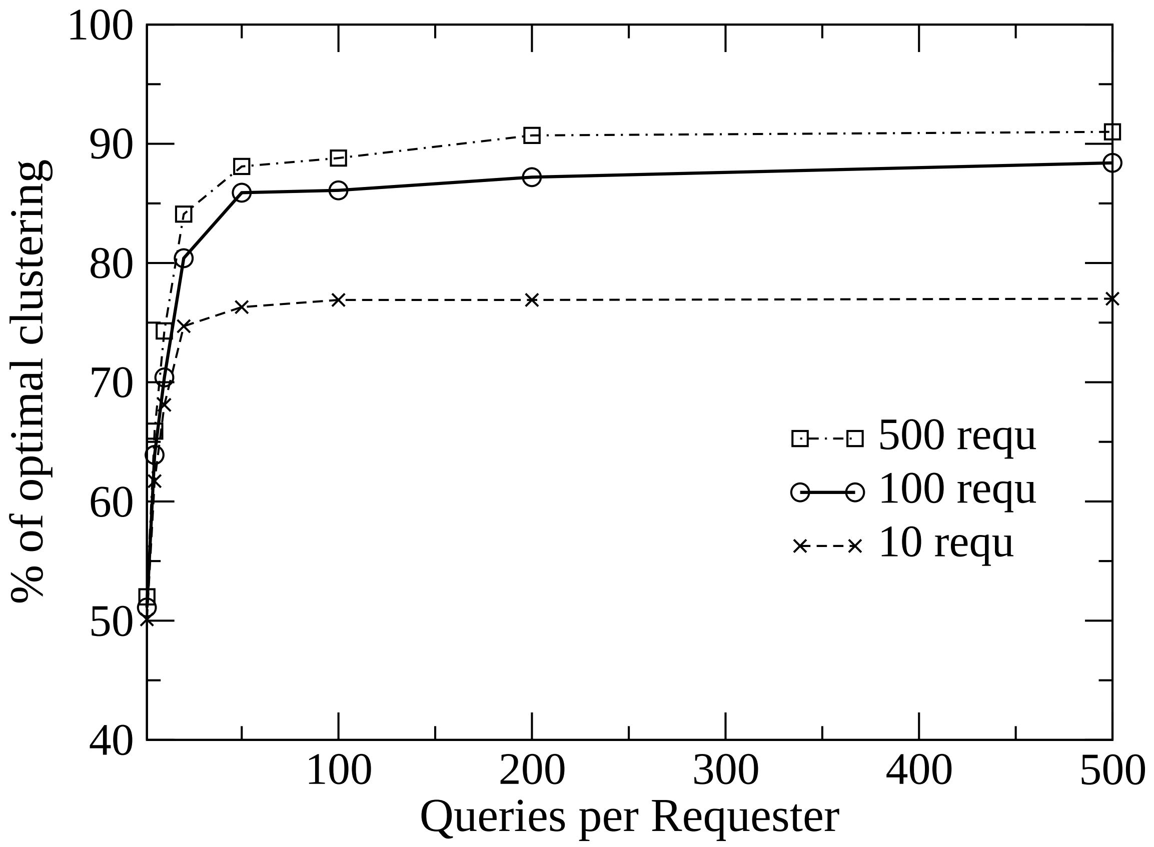
- Simulator in C
- Pure P2P model, 1-10K graphs (power-law and random)
- 100 attributes, random assignment (10 avg.)
- Queries on local schemas (3 attributes avg.)
- 100 requesters
- Compared methods
  - *Naïve rewriting (naïve)*
  - *Flooding with each peer answering Q_{orig} (Brute) – and some variations*

R



Similarity of results to the original query over variable number of queries per requester

Re



100
90
80
70
60
50
40
% of optimal clustering
100
200
300
400
500
Queries per Requester
500 requ
100 requ
10 requ

R

Related Work

- PeerDB
- Query Reformulation (Halevy et al)
- Edutella
- Chatty Web

Handling Spatial Data

Problem Definition

- Assumptions:
  - *Dissemination of spatial data in autonomous nodes*
  - *Structured flat P2P overlay*
- Problem:
  - *Necessity for indexing and routing techniques for such an environment*

Technique Requirements

We need a technique that:

- Guarantees the retrieval of any existing spatial information in the system
- Achieves satisfying index size for any node
- Achieves a satisfying length for any search path
- Provides easier access to popular information
- Preserves proximity of areas

Related Work

- VBI-tree (ICDE ‘05)
- P2PR-tree (Workshop of EDBT ‘04)
- Quad-tree approach (Poster in ICDE’05)
- kd-tree approach (WebDB’04)

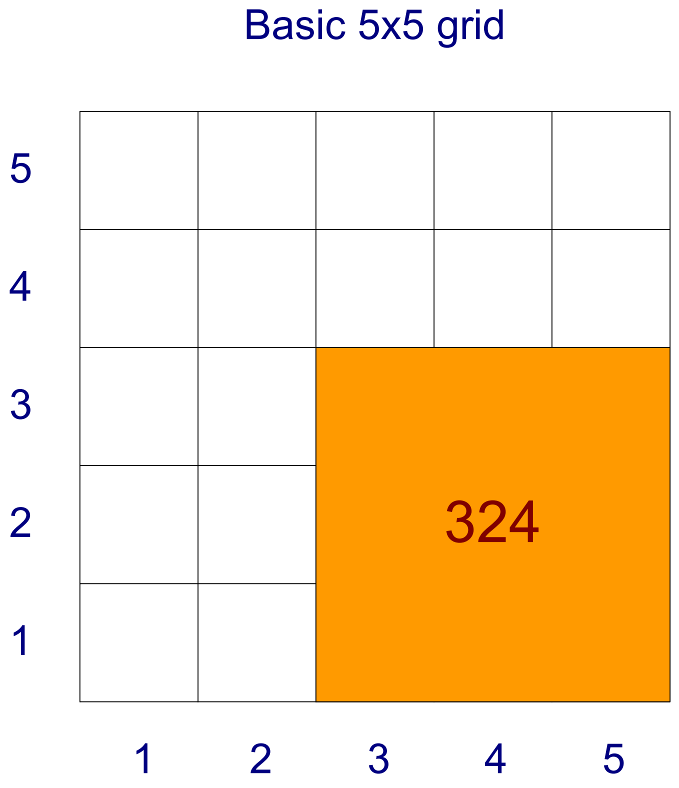
Specification of Spatial Areas

Partitioning Space:

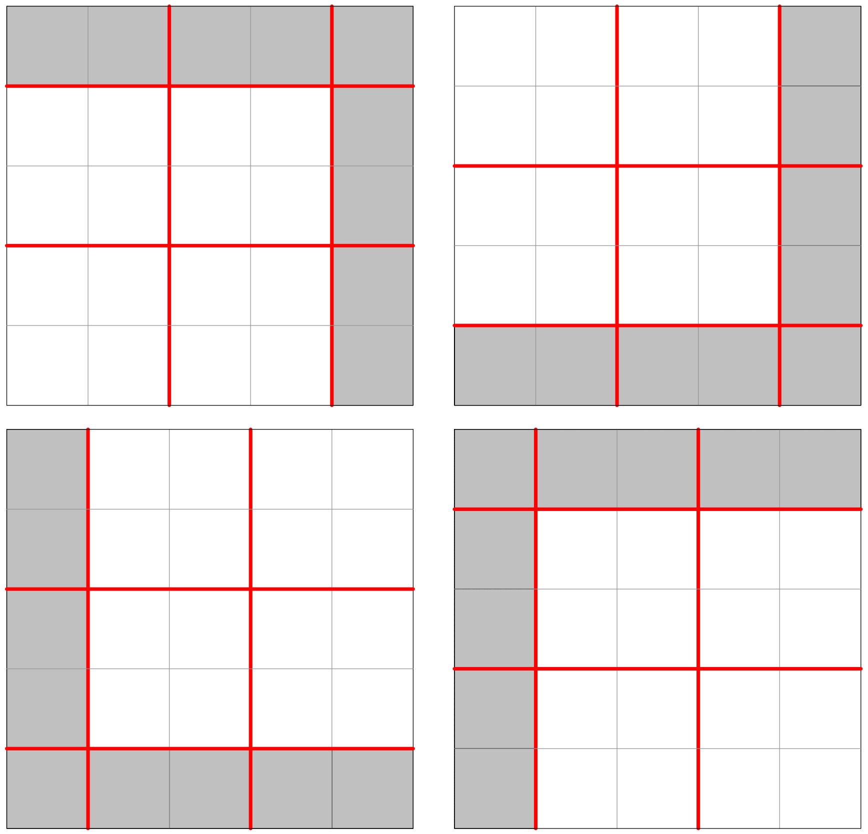
- We assume a basic fine-grained $n \times n$ grid
- Specifying Spatial Areas:
- We consider any $i \times i$ area on the basic grid, $1 \leq i \leq n$

  $\Rightarrow$ Each $i \times i$ area belongs *to* an $i \times i$ grid
- The id of an area A is: $\boxed{\mathbf{A \equiv a_z a_x a_y}}$
- $\mathbf{a_z}$: is the granularity level, i.e. the size in terms of basic cells.
- $\mathbf{a_x}$: x coordinator of the center basic cell.
- $\mathbf{a_y}$: y coordinator of the center basic cell.

Grid & Area Examples



The four 2nd level grids (z=2)



Distributed Indexing for Spatial Areas

Using Chord as the basic DHT we index areas as follows:

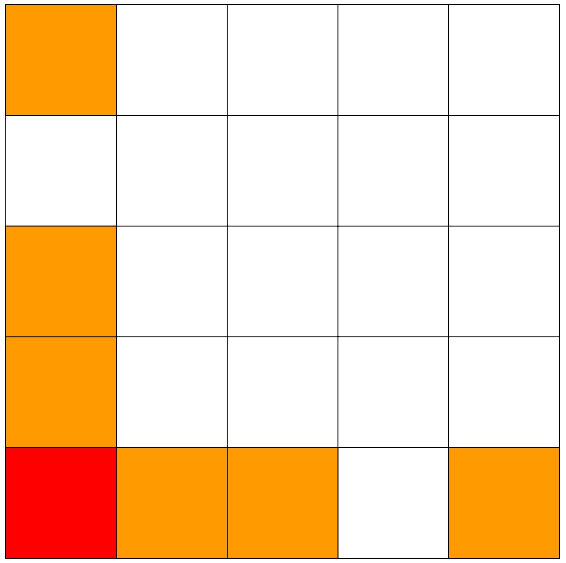
Distributed Indexing Algorithm

- **Step 1**: Indexing the grids of each level
  - *Phase A: Indexing areas of the same grid*
  - *Phase B: Indexing areas of other grids of the same level*
- **Step 2**: Indexing the grids of other levels

Indexing Examples

Example of indexing on the basic grid with Chord of base 2

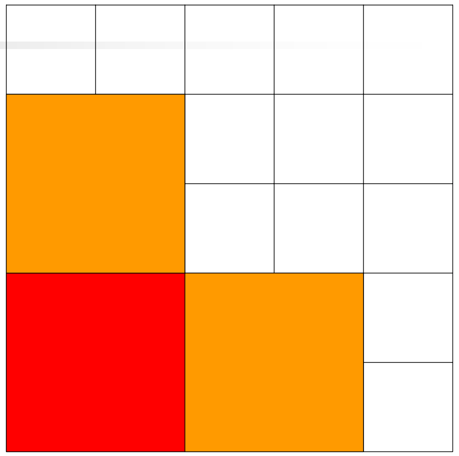
Basic 5x5 grid



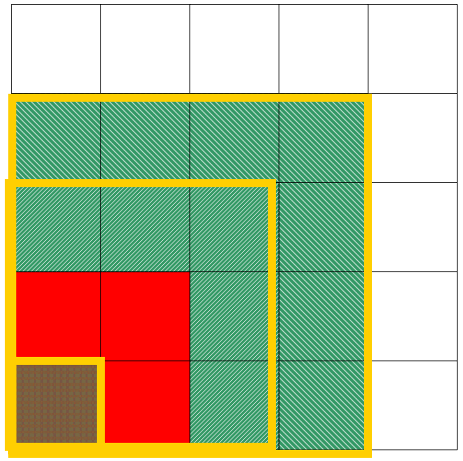
🟥 Area stored in the indexing peer

Example of indexing on a 2nd level grid with Chord of base 2

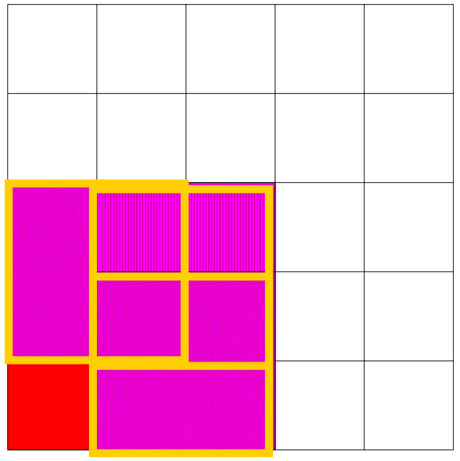
Indexing non-overlapping areas of the same size



Indexing areas of different size



Indexing overlapping areas of the same size



Direct Neighbors & Data Hashing

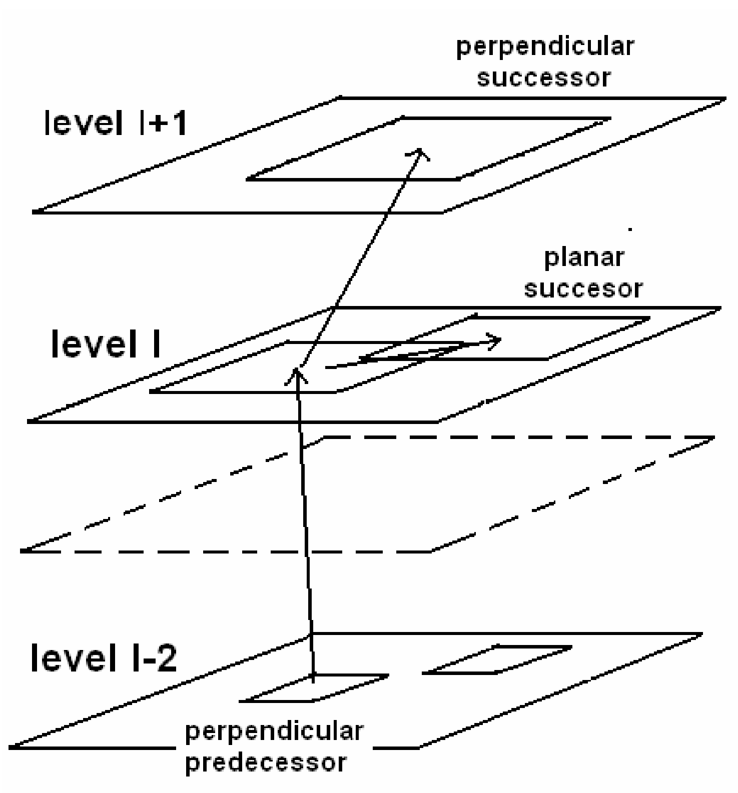
Peers have neighbors on each dimension:

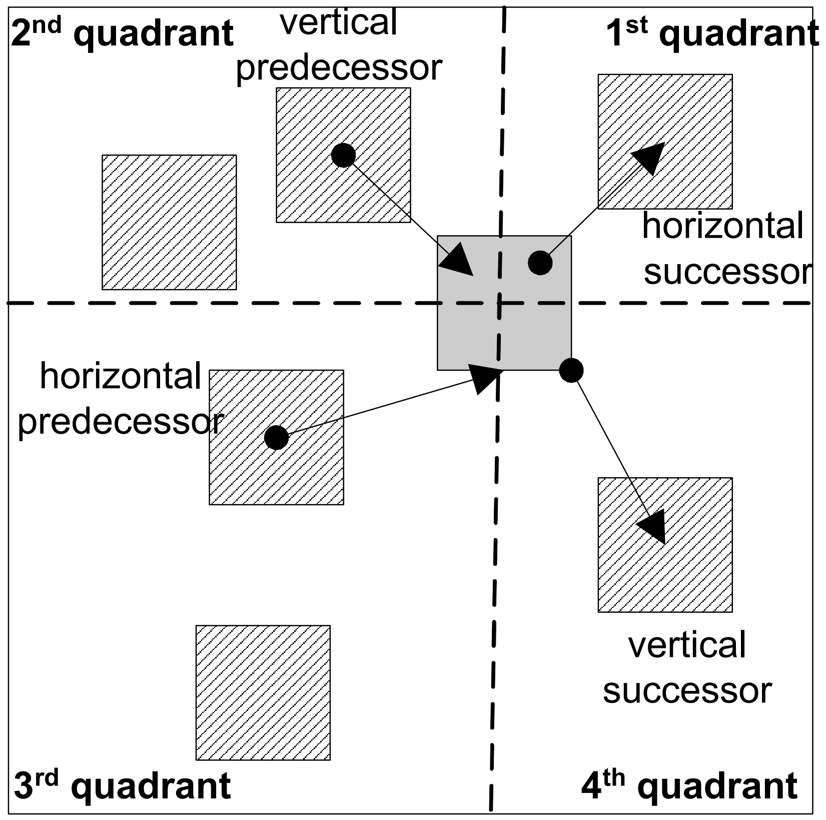
- *Horizontal*
- *Vertical*
- *Perpendicular*

Neighbors are the closest peers on each direction.

Areas are stored to the closest neighbor following a priorities of dimensions

Direct Neighbors: Examples





Routing of Spatial Queries

Basic Routing Algorithm

Step 1: Find a peer corresponding to the same size as the sought area.

Step 2: Find a peer hosting the sought area or an overlapping area of the latter

Step 3: Find the peer hosting the sought area

Complexity Characteristics

Complexity for both pure Chord and our method is $O(3\log_2 n)$.

But with our method:

- Most Important: *relatively* close areas are stored in close peers
- The search path grows with the *relative* distance
- Peers with big areas are favored with small indexes
- Shorter search paths for bigger areas

Summary

- P2P systems offer effective use of distributed resources
- Standards for P2P infrastructure are emerging
  - *Sun (JXTA) and Microsoft (.NET)*
- Content choice and control
  - *The user becomes not only a consumer but a content provider; P2P allows the end user to participate in the Internet, as the latter was originally envisioned*

AN EXCITING NEW AREA FOR DATABASE MANAGEMENT