Data Management in Peer-to-Peer Systems

Survey by A.Sung, N.Ahmed, R.Blanco, H.Li, M.Soliman, D.Hadaller

Rolando Blanco

September 21st, 2005

P2P Systems

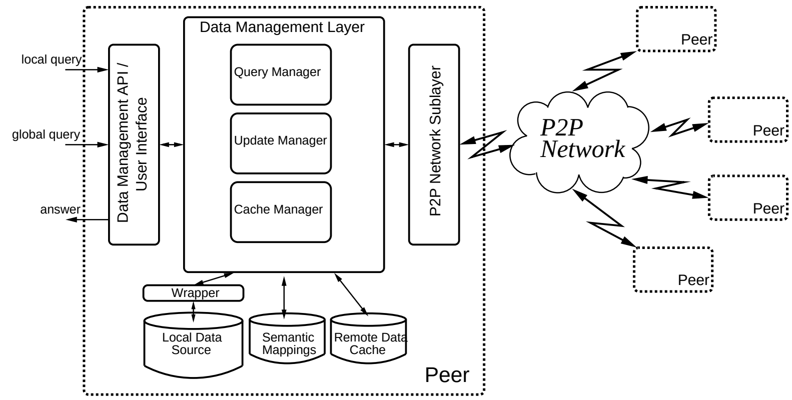
- Systems for sharing large amounts of resources
- Massively distributed
- Highly volatile
- Communication via overlay network topology
- No costly infrastructure
- Resilient to node failures
- Low overhead on participating nodes
- "Pure" P2P systems:
  - All participants (peers) have the same functionality
  - Sharing done by direct exchange

Initially for sharing unstructured data (e.g. music files). Recently proposed systems support structured data

Data Management Issues

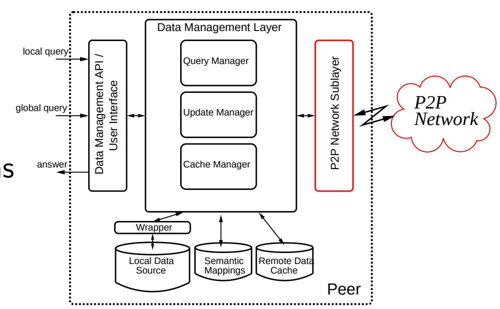
- Data location
  - Referring to/locating data in other peers
- Query processing
  - Identifying data relevant to a query
  - Efficiently executing the query
- Data Integration
  - Accessing/referring to data if different schemas/representations
- Data Consistency
  - Data replication/caching maintenance

Peer Reference Architecture



Outline

- Introduction
- → Network Structure
  - Unstructured P2P Systems
  - Structured P2P systems
- Query Processing
- Data Integration Issues
- Data Consistency Issues



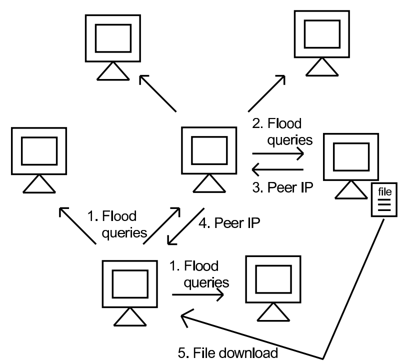
Network Structure

- Unstructured (Early) P2P Systems
  - No restriction on data placement in overlay topology
- Structured P2P Systems
  - Distributed Hash Tables (DHTs)
  - Data addressing and lookup engines: `lookup(key) = peer`
  - Overlay network topology optimized for data lookup

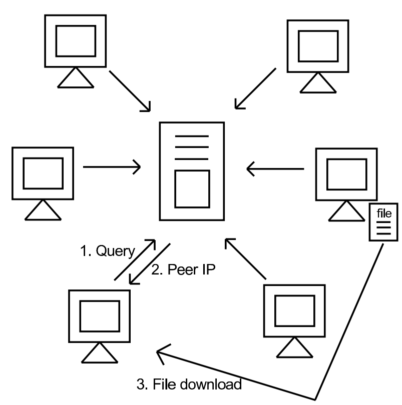
Unstructured P2P Systems

- Initially for sharing unstructured data (files)
- How to find files: index files map data files to peers
- Issue: location on the network of data files and indices
  - Pure systems
    * Each peer stores index of local data files
  - Hybrid (Client/Server) systems
    * Central server or cluster stores global index
  - Super-peer systems
    * Specialized peers store indices and/or data files

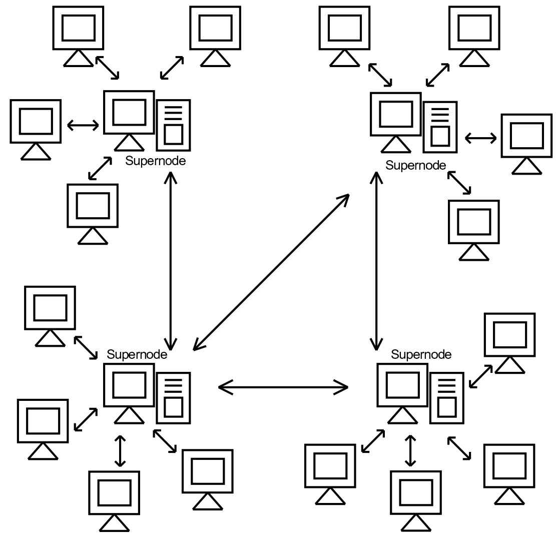
Pure Unstructured Systems



Hybrid Systems



Super-node Systems



Some Unstructured Systems

Napster	Hybrid P2P with central cluster of approximately 160 servers for all peers
Gnutella	Pure P2P
FastTrack /KaZaA	Super-nodes
eDonkey2000	Hybrid P2P with tens of servers around the world. Peers can host their own server
BitTorrent	Hybrid P2P with central servers called Tracker. Each file can be managed by a different

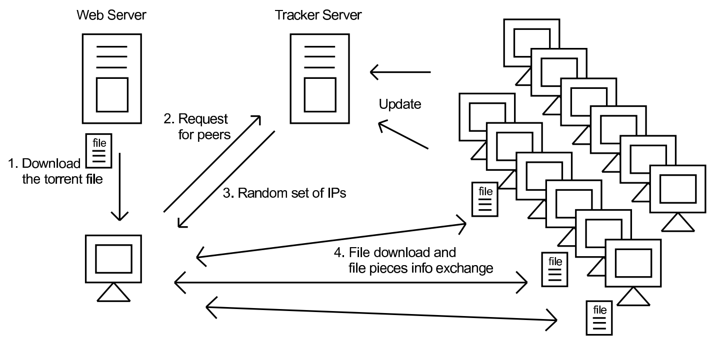
Index Management

- Napster:
  - Metadata (indices) kept in central server
  - Peers report bandwidth, number of shared files, uploads and downloads in progress, filename and size of shared files, IP.
  - Metadata uploaded when peer joins network
- Gnutella:
  - Indices stored locally
- Freenet:
  - Indices stored locally, but may not belong to peer → indices are signed.
- FastTrack /KaZaA:
  - Filename, size, modification time, content hash and file descriptors from shared folder kept in super-node

Index Management (cont'd)

Locating Data:

- *Keyword search* (routing discussed later)
- *By file identifier*: `.torrent files`
- BitTorrent



Trends

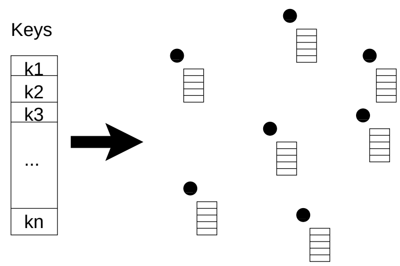
- Index storage location from central nodes, to peers, to super-peers.
- File identifiers globally unique (hashing contents)
- Sharing incentives:
  - Free-ride issue ([HCW05]) :
    * Voluntary contribution of resources
    * 66% share no files
    * 73% share 10 files or less
    * Top 1% peers accounted for 47% of query hits
    * Top 25% peers accounted for 98% of query hits
  - Anti-free riding:
    * BitTorrent: upload part of the protocol
    * Other approaches proposed include ignoring queries from free-riders, and peer expulsion.

Structured P2P Systems

- Motivated by poor scaling and recall issues in some unstructured systems
- Structured overlays: distributed hash tables (DHTs)
- Support lookup, not typically involved in actual data retrieval
  (`lookup(key) = peer`)
- DHTs implement:
  - Data and peer addressing (hash function)
  - Routing protocol and overlay structure
  - Routing state maintenance protocol

Data and Peer Addressing

- Goal: uniform distribution of keys in the overlay
- Some systems implement non-uniform hashing to favour locality/proximity or certain types of queries (e.g. range queries)
- SHA-1 (Secure Hash Algorithm) most widely used base hash function
- Hashing assigns keys to peers



Routing Protocol

- Peers do naming (assign ids) and routing
- Provides location of peer assigned to hashed key
- Implements lookup functionality (`lookup(key) = node`)
- Implementation closely associated with overlay structure used
  - Ring: Chord
  - Hypercube: CAN
  - Tree: Tapestry, ...
  - Hybrid: Pastry (Tree/Ring), ...
  - ...
- Routing protocols attempt to provide efficient lookups and minimize routing state

Routing State Maintenance Protocol

- Routing protocol requires peers to maintain routing state
- Routing info differs between routing protocols
- Maintenance algorithms to keep routing state up-to-date and consistent.
- *Issue:* Peer volatility (churn), undependable networking

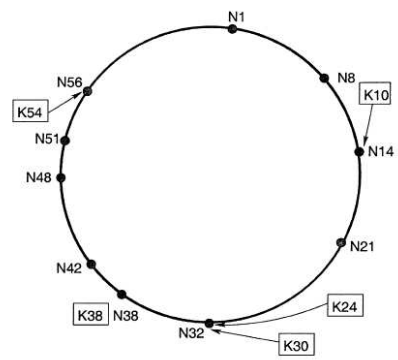
Chord

[SMLN+01]

- `lookup(key) = IP address`
- Peers IPs and keys assigned a *m-bit* identifier using SHA-1
- *m* should big enough so that chance of two keys/IPs having the same hashing is negligible
- Identifiers ordered in a circle module 2^m (*Chord Ring*).
- Key k assigned to first peer whose identifier is equal or follows k's identifier

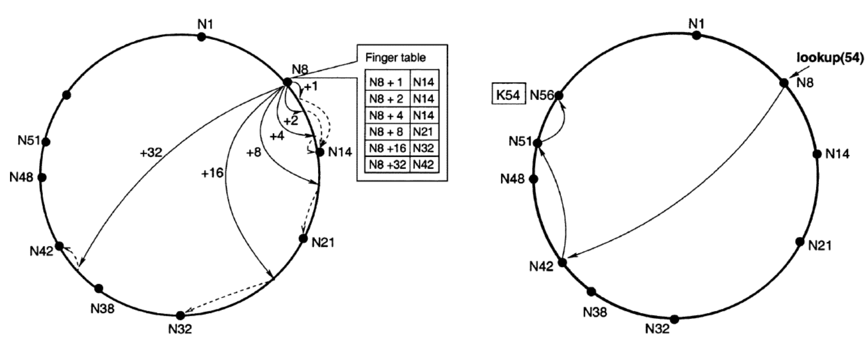
Chord Ring Example

`m = 6`

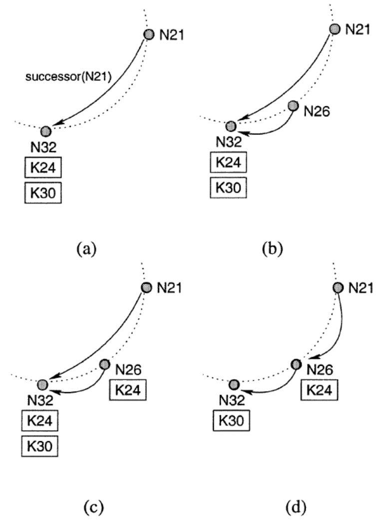


Chord Lookup Example

`lookup(K54)`



Chord Ring Maintenance



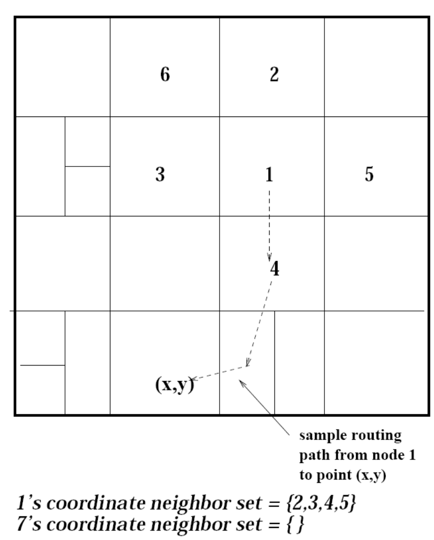
CAN: Content Addressable Network

[RFHK01]

- Virtual dimensional coordinate space
- Each peer holds a zone
- Key is hashed to a zone, data value is stored in that zone
- `lookup(key) = value`
- Peers keep routing table with IP and virtual coordinate zones of its immediate neighbours
- Two nodes are neighbours if their coordinate spans overlap along d - 1 dimensions and abut along one dimension (terminate at a point of contact)
- Routing by choosing closest neighbour in direction to destination

CAN Example

2d space

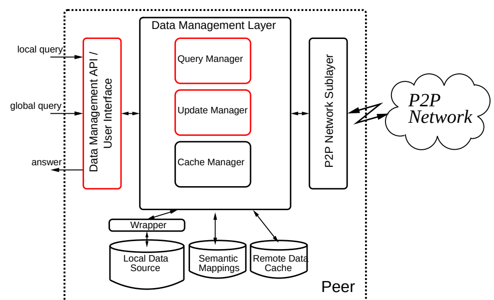


Issues

- DHTs:
  - Scalable and decentralized
  - Uniform distribution of data (*keys*)
- Issues:
  - Data may be located far away from users (no control over data placement)
  - Neighbour peer in overlay may be far away in underlying network
  - Hotspots (popular data)
  - Lookup service only. `Select ... from ... where key = value`; what about range queries and aggregation ?
- Several proposals to address these issues (e.g. SkipNets)
- Still, no widespread use

Outline

- Introduction
- Network Structure
- → Query Processing
  - File Sharing Systems
  - Querying Structured Data
- Data Integration Issues
- Data Consistency Issues



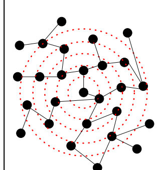
File Sharing P2P Systems

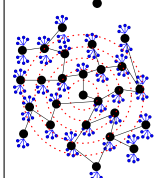
- Purpose is to locate peers that store a requested file
- In contrast to DHTs, files generally not (uniformly) redistributed.
- Most requested files usually replicated at many peers
- Unpopular files in the system may not be found (contrast with perfect recall in DHTs).
- Early systems use centralized indexing server
- More recent systems route queries to peers in overlay based on some routing scheme

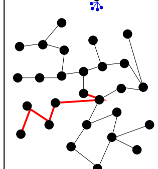
Query Routing

- Blind search
  - Query arbitrarily forwarded to peers
  - No guarantee peer receiving query can satisfy query
  - Simple, minimal or no metadata required
- Informed search
  - Query forwarded to peers that may satisfy query with high probability
  - Additional knowledge needed to route queries
  - Considerable amount of metadata required

Blind Search







- No information stored regarding data placement
- Flooding:
  - Query sent to all neighbours
  - Time-to-live (TTL) used to limit radius of flood
  - Can overload network quickly
- Improvements:
  - Super-nodes acting as proxys
    * Flooding between super-nodes only
    * Poor scalability as number of super-nodes grows
  - Random walk
    * Query forwarded to one neighbour at a time + TTL
    * Reduction in traffic
    * Searches may take longer
    * Popular files still found quickly

Informed Search

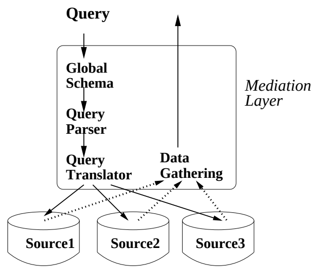
- Goal is to increase probability of locating data item as hop count increases
- Query Routing Protocol:
  - Keywords describing file contents summarized in bloom filters
  - Bloom filters are propagated for a number of hops
  - Queries propagated to neighbours if keywords match
  - Blind search if no matches
- Scalable Query Routing
  - Random walk + bloom filters with information that decreases exponentially as the hop count increases.

Informed Search (cont'd)

- Adaptive Searches
  - Counter kept for each (file, neighbour) combination
  - Counter incremented if forwarding neighbour finds file
  - Issues:
    * Performs well if peers in system for long periods
    * Hotspots since paths gain popularity quickly; under utilization of other possible paths
- Several other variations ...

Querying Structured Data

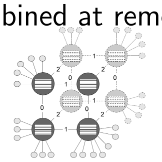
- Traditional centralized approaches
  - Mediation layer:



    * Provides global schema
    * Decomposes queries into sub-queries
    * Gathers and combines results
  - Integrating and materializing data in a warehouse type setting
- Issues:
  - Scalability, volatility of data sources, single point of failure
- P2P approach: decentralized query processing techniques

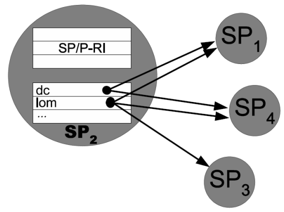
Common Data Model Systems

- Usually a distributed catalogue drives the distribution of queries among peers based on query type, attributes and/or data filters
- Edutella ([NWQ+02]):
  - Unstructured super-peer system
  - Common query exchange language
  - Support for limited amount of data representations
  - SP-to-SP and SP-to-Peer indices at super-peers
  - Query routed first at super-peer level and then distributed to peers
  - Results are combined at remote peers



Common Data Model - Examples

Edutella

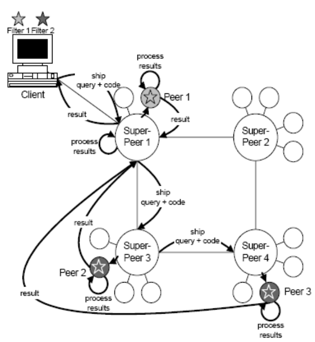


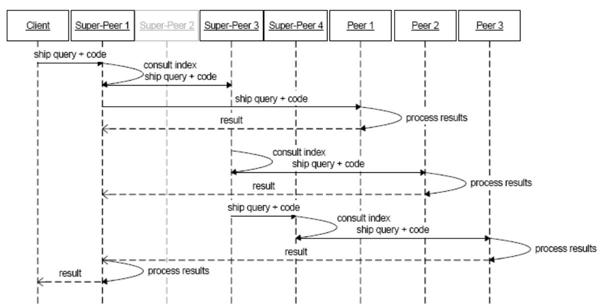
Granularity	Index of SP_2		
Schema	dc lom		SP_1, SP_3, SP_4 SP_1, SP_4
Property	dc:subject dc:language lom:context		SP_1, SP_3, SP_4 SP_1, SP_4 SP_1, SP_4
Property Value Range	dc:subject dc:subject	ccs:networks ccs:software-engineering	SP_3 SP_1, SP_4
Property Value	lom:context dc:language	"undergrad" "de"	SP_1, SP_4 SP_1, SP_4

- User defined code can be pushed to peers for execution
- Clustering of peers with common data

Common Data Model - Examples

Edutella ([BDK+03])





Common Data Model - Examples

Galanis et al. [GWJD03]

- Idea:
  - Index (`key, node`) pairs
  - Index points to structural (XML) and value summary of data in `node`
- Given a query $a_1/a_2/.../a_n/k\ op\ x$:
  - Retrieve summaries
  - Use structural summary to decide if $a_1/a_2/.../a_n/k$
  - Use value summary to decide if $k\ op\ x$
- Implementation:
  - Distributed catalogue using DHT (Chord)
  - DHT keys: $k's$; DHT values: node (peer) where summary for k is stored
  - Summary locally implemented as B+ tree

Common Data Model - Examples

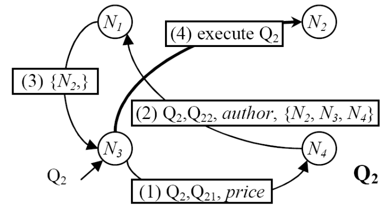
Paths in XML Data	
N_1	library/catalogs/book/author, library/reservation/book/author
N_2	bookstore/book/price, bookstore/book/author
N_3	bookstore/book/price, bookstore/book/author
N_4	bookstore/book/price, bookstore/book/author

Table 1: Nodes with sample data

DHT Index	
N_1	author: {($S_{1,author}$), ($S_{2,author}$), ($S_{3,author}$), ($S_{4,author}$)}
N_2	reservation: {($S_{1,res.}$)}
N_3	--
N_4	book: {($S_{1,book}$), ($S_{2,book}$), ($S_{3,book}$), ($S_{4,book}$)}, price: {($S_{2,price}$), ($S_{3,price}$), ($S_{4,price}$)}

Table 2: Part of the DHT index on each node

Query: Q_2: //book[author = "J Smith"]/price on N_3
N_3: query_parts(Q_2) = Q_{21}: //book/**price**
Q_{22}: //book/**author** = "J Smith"
N_3: dht::lookup(price) = {N_4}
Q_2 and Q_{21} sent to N_4
N_4: map(price, //book/price) = { N_2, N_3, N_4 } (B+ tree)
dht::lookup(author) = { N_1}
Q_2, Q_{22}, {N_2, N_3, N_4} sent to N_1
N_1: (map(author, /book/author) and *author* = *"J. Smith"*) = {N_2}
{N_2, N_3, N_4} $\cap$ {N_2} = {N_2}
{N_2} sent to N_3
N_3: sends Q_2 to N_2
N_2: executes Q_2 and returns results to N_3



Heterogeneous Data Model Systems

- Peers need to apply data integration techniques to identify content or structural similarities among peers
- Peer selection is a big issue
- Common approaches:
  - Data translation rules to global schema or among peers
  - Gossiping, Information retrieval, semantic mappings
  - Query reformulated to match other peers' schemas
  - Data merging at super peers or originating peer
  - Some systems require human intervention

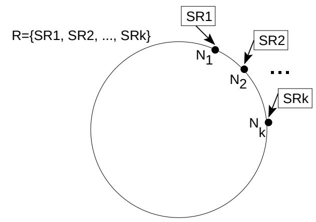
More on this later

Complex Queries - Range Queries

- Structured P2P Systems (DHTs):
  - Major problem: randomizing hash function
  - Approaches include hashing ranges instead of single values, range-aware hashing, augmenting metadata information
  - Introduction of load balancing problems, and sometimes a-priori knowledge of interesting ranges
- Non DHT solutions:
  - Most proposals consist in extending distributed catalogue

Complex Queries - Multi-Attributes

- MAAN Multi-Attribute Addressable Network [CF03]



  - Built on Chord; Supports multi-attribute and range queries
  - Numeric attributes hashed with locality preserving hash function ($if\ n_1 > n_2\ then\ h(n_1) > h(n_2)$)
  - Hashing function parameter is pair $(attr, val)$
  - Multi-attribute query split into single-attribute queries
  - Sub-queries are executed and merged at query originator
  - Query selectivity breakpoint at which flooding is more efficient

Complex Queries - Joins

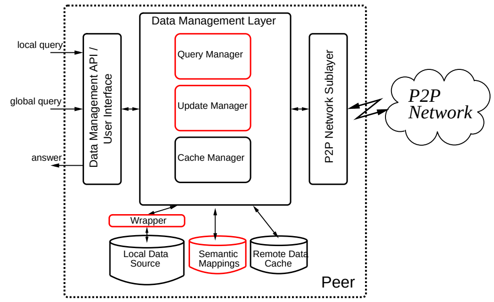
- PIER [HHL+03]
  - DHT (CAN) based
  - Key constructed from a "namespace" (relation) and "resourceID" (primary key)
  - Queries multicasted to all nodes in namespace to be joined
    * Symmetric hash join: each node in each namespace hashes tuples into new namespace
    * Fetch matches: one of the two namespaces already hashed on the join attribute. Each node in the second namespace finds the matching tuples from the first namespace using DHT get operations

Complex Queries - Ranking

- Top-k in Edutella:
  - Uniform schema and ranking function at peers
  - Peers evaluate top-k query locally, send results and scores to super-peers
  - Super-peers select results with highest scores

Outline

- Introduction
- Network Structure
- Query Processing
- → Data Integration Issues
  - Traditional Approach
  - Schema Mappings
  - Data Mappings
- Data Consistency Issues



Data Integration - Example

University of Waterloo (UW):

- Areas(area_id, name, description)
- Projects(proj_id, area_id, name)
- Publications(pub_id, title, year, howpublished)
- AuthorPublications(author_id, pub_id)
- ProjectPublications(proj_id, pub_id)
- Researcher(res_id, name)
- ProjectMembers(res_id, proj_id, role)

University of Toronto (UT):

- Project(projID, name, desc)
- Student(studID, fname, lname, status)
- Faculty(facID, fname, lname, rank, office)
- ProjMember(projID, memberID)
- Paper(paperID, title, forum, year)
- Author(authorID, paperID)

The University of British Columbia (UBC):

- Area(areaId, name, descr)
- Project(projName, sponsor)
- ProjArea(projName, areaID)
- Pubs(pubID, projName, title, venue, year)
- Author(pubID, author)
- Member(projName, member)

...

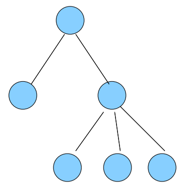
Data Integration

- Semantically, databases store same type of data
- Peers should be able to expose only the portions that they want to contribute to the system
- *Semantic mappings*: describe the relationships between two or more schemas for the purpose of sharing and integrating data:
  - Schema mappings
    - Specify equivalence between relations and attributes
    - Used when different names or formalisms to represent data
  - Data mappings
    - Specify equivalence between attribute values
    - Used when semantic differences between schemas make schema mapping inapplicable
- Schema and data mappings complement each other

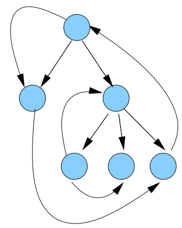
Traditional vs P2P

"... the true novelty lies in the PDMS ability to exploit transitive relationships among peers' schemas" [HIST04]

From:



To:



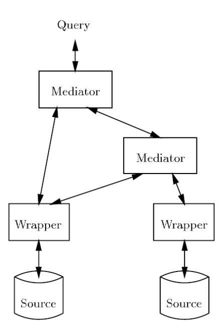
Schema Mappings

{
UW.Projects.proj_id ⟼ UT.ProjectID,
UW.Researcher.name ⟼ concat(UT.Faculty.fname, ' ', UT.Faculty.lname),
UW.Researcher.name ⟼ concat(UT.Student.fname, ' ', UT.Student.lname)
}

- Transitive; may or may not be one-to-one and reflexive
- Purpose is to provide uniform querying environment that hides heterogeneity and distribution
- Typically manually specified, several approaches proposed for automating

Traditional Approach

- Mediated schema between data sources



[Ull97]

- Provides global unified schema
- Queries specified in terms of global schema
- Query reformulated based on schema mappings
- Wrappers close to sources provide translation services if required (to deal with local query language, model or any other incompatible feature)
- *Semantic tree*: Mediated schemas can be constructed based on other mediated schemas

Traditional Approach (cont'd)

- Mediated schema between data sources (cont'd)
  - Strategies for the definition of schema mappings between mediated and local schemas:
    * Global-as-view (GAV)
      $G = F(L_1, L_2, ...)$
    * Local-as-view (LAV)
      $L = F(G_1, G_2, ...)$
    * Global-and-local-as-view (GLAV)
      $F(G_1, G_2, ...) = F'(L_1, L_2, ...)$

Applicability in P2P Systems

- Volatility: as peers join/leave system mediated schema needs to be updated
- Peer autonomy: some participants may be willing to contribute only a portion of their data
- Scalability: where is the global schema stored: centralized, distributed, replicated?

Unique global schema seems impractical in P2P systems, instead:

- Pair mappings
- Peer-mediated mappings
- Super-peer mediated mappings

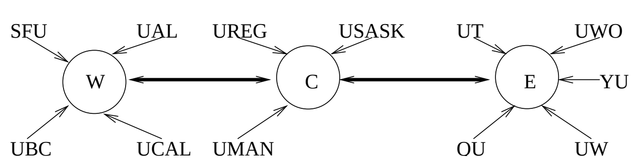
Schema Mappings in P2P Systems

UBC $\longmapsto$ UW $\longmapsto$ UT
map(UW) map(UT)

(a) Pair Mappings

UBC $\longleftarrow$ UW $\longrightarrow$ UT
map(UW, UT, UBC)

(b) Peer Mediated Mappings



(c) Super-peer Mediated Mappings

Schema Mapping Maintenance

- Machine learning techniques

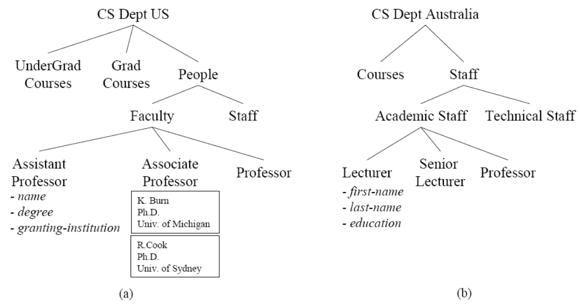


Fig. 1 Computer Science Department Ontologies.

[DMD+03]

- Idea: given taxonomies of two different ontologies, find similar concepts
- Concept classifiers: to decide similarity between concepts A and B, data for B classified with A's classifier and vice-versa. Amount of values that can be successfully classified into A and B represent the similarity between A and B.

Schema Mapping Maintenance (cont'd)

- Common agreement mappings:
  - Schema mapping done between peers with common interest for sharing data
  - Manually maintained by privileged or expert users
- Semantic gossiping
  - Initial common agreement mappings built by experts
  - Queries propagated towards peers without direct mapping
  - Semantic agreement measured by analyzing results
  - New mappings created or old mappings adjusted

Schema Mapping Maintenance (cont'd)

- Information Retrieval Approaches

Peer	Names	Keywords
P1	Kinases	protein, human
	SeqID	key, identifier, ID
	length	length
	proteinSeq	sequence, protein sequence
P2	Protein	protein, annexin, zebrafish
	SeqNo	number, identifier
	len	length
	sequence	sequence
P3	ProteinKLen	protein, kinases, length
	ID	number, identifier
	seqLength	length
	ProteinKSeq	protein, sequence
	ID	number, identifier
	sequence	sequence
P4	Protein	protein, kinases, annexin, ...
	name	name
	char	characteristics, features, functions

- Descriptive words for attributes/relations
- Queries flooded to neighbouring peers
- Peers receiving query, decide if matching attributes in local schema using IR techniques
- User confirms matching, system remembers

[NOTZ03]

Data Mappings

- Schema mappings work well when schema differences are mainly structural
- When attribute values differ but are semantically related, data mappings are used
- Implemented as relations (*mapping tables*) on the attributes being mapped
- Created by experts, some proposals to validate mappings
- Very common integration method in real-world applications
- Data mappings define semantic graph among peers as well
- Not as much work done as with schema mappings

Data Mappings (cont'd)

GDB_id	Gene Name
GDB:120231	NF1
GDB:120232	NF2
GDB:120233	NGFB

4(a) Relation in GDB

SwissProt_id	Protein Name
P21359	NF1
P35240	MERL

4(b) Relation in SwissProt

GDB_id	SwissProt_id
GDB:120232	P35240
v – { GDB:120232 }	v' – { P35240 }

4(c) Mapping table from GDB to SwissProt

GDB_id	Gene Name	SwissProt_id	Protein Name
GDB:120231	NF1	P21359	NF1
GDB:120232	NF2	P35240	MERL
GDB:120233	NGFB	P21359	NF1

Tuples that can be mapped from GDB to SwissProt

[KAM03]

- Specification of different semantics for data mappings
- Work proposed to validate mappings and to infer new mappings

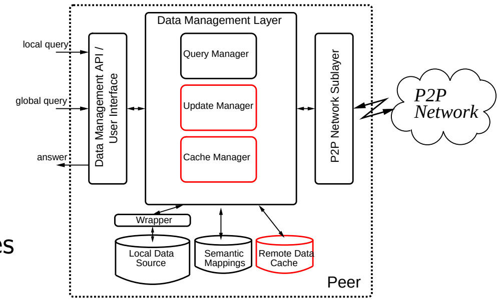
Current Research

- Mapping compositions
- Event/trigger mechanisms to enforce data mappings or propagate data among peers

Outline

- Introduction
- Network Structure
- Query Processing
- Data Integration Issues

→ Data Consistency Issues
  - P2P Specific Challenges
  - Solutions



Data Consistency Issues

- Arise in any scenario involving data duplication
- Caching
  - Commonly done for performance
  - There is a single authoritative source for the document
- Replication
  - Done for availability, performance, ...
  - All replicas regarded as equal

P2P Specific Challenges

Challenge	**Implications**
High churn rate: nodes frequently joining, leaving, and failing	Must have ways of maintaining the network structure, e.g. using a DHT
Lack of global knowledge	Must act on partial knowledge, such as probabilistic measures
Low online probability	Peers are offline most of the time and cannot be relied on to keep data intact
Unknown and varying node capacity	Can't assume well connected and powerful infrastructure; must be sensitive to individual capacity
Overlay topology is independent of physical topology	One hop in the overlay may be a large physical distance; must be aware of underlying topology

Solutions

Challenge	Solutions in Replication	Solutions in Caching
High churn rate and low online probability (*how to guarantee data is not lost; data consistency and availability*)	Must maintain k online replicas: (1) Using estimated global information and probabilistic methods, (2) Store k replicas at k successors in a Chord ring, (3) Maintain replicas at nodes with k closest numeric Ids in the DHT	No need to maintain availability of cached data, use DHT to maintain lookup; peers cache whatever is available
Lack of global knowledge (*data location*)	(1) Use estimated global information based on rumor spreading, (2) Not an issue if using a DHT	Global knowledge would help optimize cache placement, but not necessary
Unknown and varying node capacity	(1) Often ignored, (2) Nodes advertise their capacity, replica placement is based on these capacities	Cache and share only what you choose

Solutions (cont'd)

Challenge	Solutions in Replication	Solutions in Caching
Overlay topology is independent of physical topology	(1) Often ignored, (2) Use a DHT with locality properties	(1) Often ignored, (2) Use a DHT with locality properties
Updating replicas / Reducing staleness	(1) Push updates to all active replias, (2) Pull updates from most recent replica when required (3) Store updates as node-specific log entries	(1) Assume data doesn't go stale, (2) Web: Expiry (TTL), Conditional GETs (If-modified-since)

Example: Support for Range Queries

`http://www.csg/~rblanco/w05/cs856/report.pdf`

Assumptions:

- High semantic similarity between data sources in the system
- Very large number of peers
- Location of the data cannot be altered
- A high percentage of queries are range queries
- Range conditions are specified for a small subset of attributes

Example: Support for Range Queries

Assumptions (cont'd)

- Attribute updates happen no more than a few times a week
- Peers providing data have an incentive to participate in the system
- Best effort, good enough answers are acceptable

Common approach:

- Use of structural information (attribute occurrence)

```
SELECT product_dim, prod_cost
  FROM products
 WHERE product_class = 'TV'
   AND product_unit = 'INCHES'
   AND product_dim BETWEEN 17 AND 21
   AND product_cost BETWEEN 100 AND 200
```

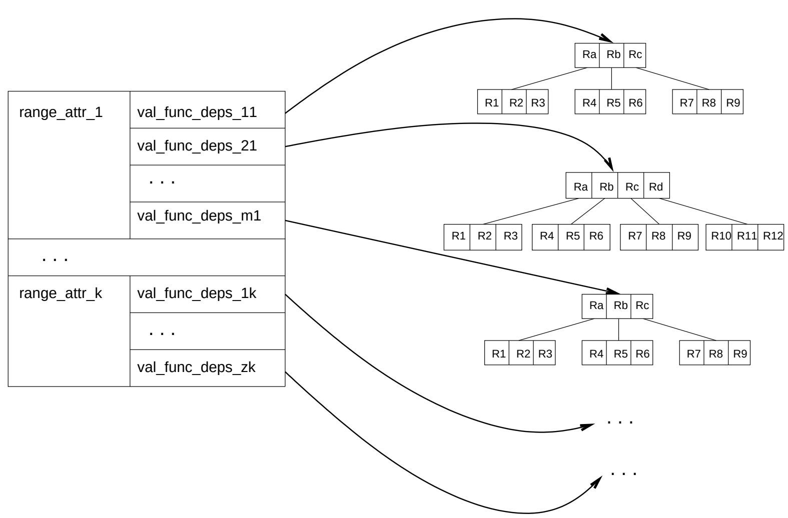

Example: Support for Range Queries

Proposal:

- Peers register:
  - Schema descriptions with synonyms/alternate names (schemas assumes to be semantically very close)
  - Range information including relevant functional dependencies
- System maintains:
  - SP-to-P and SP-to-SP catalogues with schema and range information

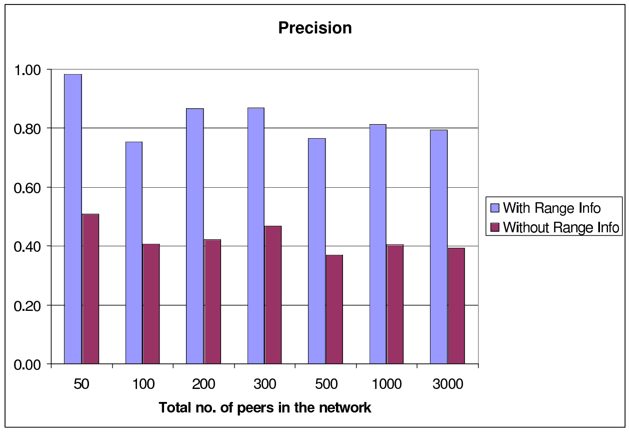
Example: Support for Range Queries

Proposal (cont'd):



Example: Support for Range Queries

Results:



References

- CS856 - Data Management in P2P Systems Survey
`http://www.csg/~rblanco/w05/cs856/p2p_survey.pdf`
- [HCW05] D. Hughes, G. Coulson, and J. Walkerdine. Free riding on gnutella revisited: the bell tolls? Draft at http://www.comp.lancs.ac.uk/computing/users/hughesdr/papers/freeriding.pdf, February 2005
- [BDK+03] Ingo Brunkhorst, Hadhami Dhraief, Alfons Kemper, Wolfgang Nejdl, and Christian Wiesner. Distributed Queries and Query Optimization in Schema-Based P2P-Systems. In *Proceedings of the 1st International Workshop on Databases, Information Systems, and Peer-to-Peer Computing (DBISP2P 2003)*, pages 184–199, 2003
- [CF03] M. Cai and M. Frank. MAAN: a Multi-Attribute Addressable Network for Grid Information Services. In *Proceedings of the International Workshop on Grid Computing*, 2003
- [DMD+03] AnHai Doan, Jayant Madhavan, Robin Dhamankar, Pedro Domingos, and Alon Halevy. Learning to match ontologies on the semantic web. *The VLDB Journal*, 12(4):303–319, 2003
- [GWJD03] L. Galanis, Y. Wang, R. Jeffery, and DeWitt D. Processing Queries in a Large Peer-to-Peer System. In *Proceedings of Conference on Advanced Information Systems Engineering (CAiSE)*, 2003

References (cont'd)

- [HIST04] Alon Y. Halevy, Zachary G. Ives, Dan Suciu, and Igor Tatarivov. Schema mediation for large-scale semantic data sharing. *VLDB Journal*, 2004
- [HHL+03] R. Huebsch, J. Hellerstein, B. Lanham, Loo S., and I. Stoica. Querying the Internet with PIER . In *Proceedings of the 29th Int. Conf. on Very Large Data Bases (VLDB)*, 2003
- [KAM03] Anastasios Kementsietsidis, Marcelo Arenas, and Renée J. Miller. Mapping data in peer-to-peer systems: Semantics and algorithmic issues. In *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 325–336. ACM Press, 2003
- [NWQ+02] Wolfgang Nejdl, Boris Wolf, Changtao Qu, Stefan Decker, Michael Sintek, Ambjörn Naeve, Mikael Nilsson, Matthias Palmér, and Tore Risch. Edutella: a p2p networking infrastructure based on rdf. In *WWW '02: Proceedings of the eleventh international conference on World Wide Web*, pages 604–615, New York, NY, USA, 2002. ACM Press
- [NOTZ03] Wee Siong Ng, Beng Chin Ooi, Kian-Lee Tan, and Aoying Zhou. Peerdb: A p2p-based system for distributed data sharing. In *Intl. Conf. on Data Engineering (ICDE)*, 2003

References (cont'd)

- [RFHK01] S. Ratnasamy, P. Francis, M. Handley, and R. Karp. A scalable content-addressable network. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*. ACM Press, 2001
- [SMLN$^+$01] I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup protocol for internet applications. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*. ACM Press, 2001
- [Ull97] Jeffrey D. Ullman. Information integration using logical views. In *Proceedings of the 6th Int. Conf. on Database Theory (ICDT-97)*, 1997. Lecture Notes in Computer Science, pages 19–40. Springer-Verlag, 1997