P2P (Semantic) Mediation

Georges Gardarin
Dragan Florin, Laurent Yeh
PRiSM Lab.
University of Versailles, France

Outline

- Mediation Systems
- An example: XLive
- P2P Mediation
- An example: PathFinder
- Conclusion

1. Mediation: Motivations

- Information systems nowadays:
  - Large number of distributed and heterogeneous data sources
  - Applications must access any data easily, efficiently, securely
  - Uniform, simple, transparent, standard query interfaces needed

- Fundamentals objectives of a data mediation system:
  - Integration: Build semantic views from multiple data sources
  - Queryability: Provide a rich query and update language
  - Efficiency: Process distributed query efficiently in real time
  - Delegation: Process queries as much as possible at data source
  - Openness: Facilitate registration and withdrawal of a data source

Mediation: Data Heterogeneity

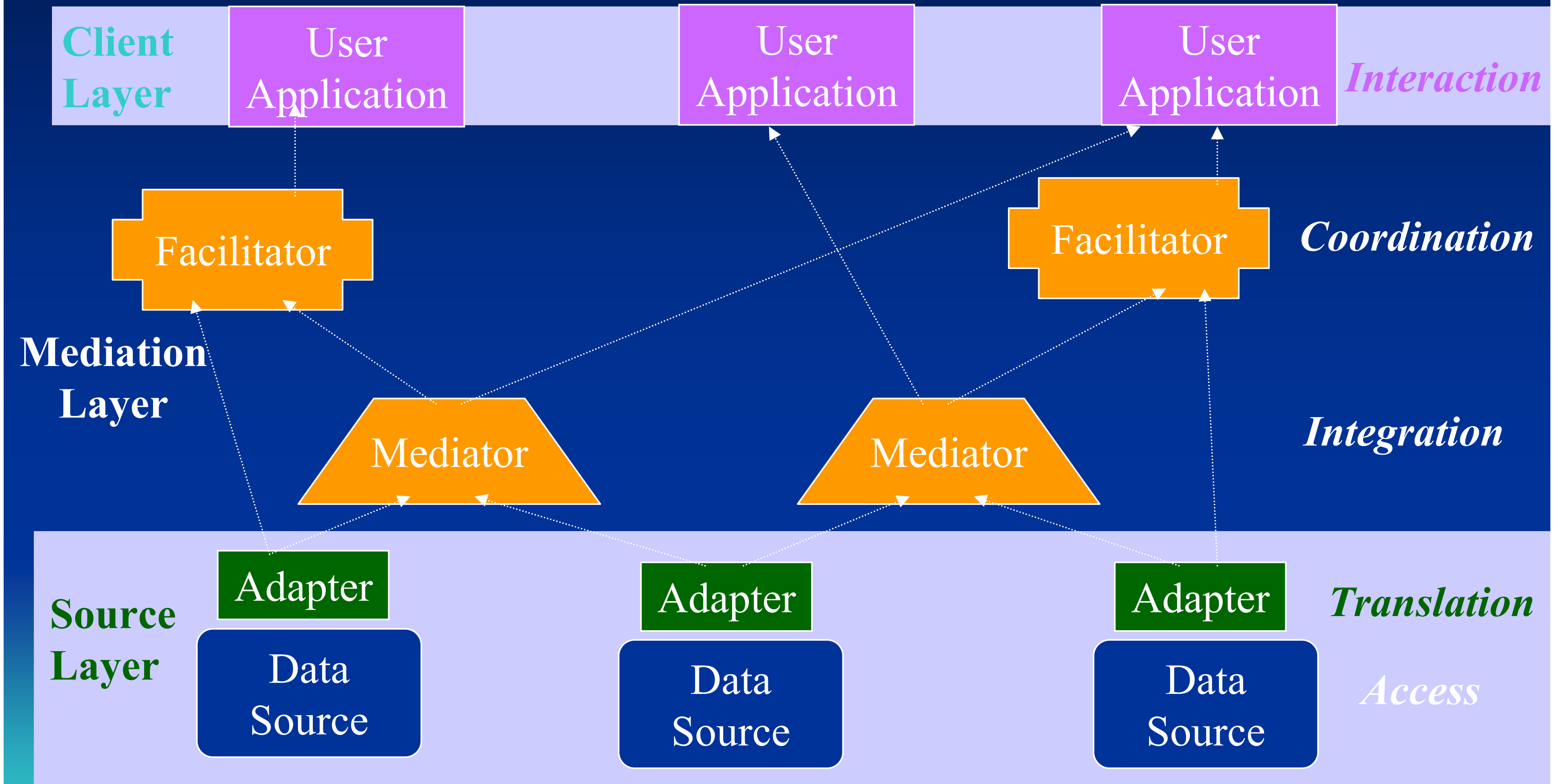
- Physical level
  - Query language: SQL, OQL, LDAP Query, XQuery
  - Result format: array of tuples, Web pages, XML documents
- Logical level
  - Simple types mismatch: address varchar (64) or unlimited string
  - Complex type mismatch: Person (Name, Firstname, Address) versus Person (SSN, Name, Street, City, zip)
- Semantic level
  - Same name to designate different things
  - Different names to designate similar things
  - Different basis, measurement units, …

Mediation: Data Distribution

- Localization of relevant data sources for a query
  - Several sources provide data for a given “semantic” concept
  - Meta-data describing the source are often used
  - The number of sources can be very large considering the web
  - Copies and redundant data should be identified and removed

- Integration of data sources with different capabilities
  - Functions can be different
    - e.g. simple keyword selection versus complex document join
  - Processing times can be different
    - e.g. mobile computer versus parallel server

Data Mediation: I3 Architecture



Data Mediation: A Long History

- Relational generation (1978-1990)
  - Centered around a relational DBMS that acts as a mediator
  - SDD1, Sirius Delta, R*, Ingres/Star, Oracle*
  - Mermaid, Multibase, Data Joiner
- Object-Relational generation (1990-2000)
  - Federate heterogeneous DBMSs around SQL3
  - Pegasus, IRO-DB, OLE-DB, Garlic
  - SQL/XML: Medience (BO), Information Integrator (IBM), OLE-DB.NET (MS), OpenLink
- XML generation (2000- …)
  - Xquare Fusion, XLive (PRiSM), Nimble (Actuate), Enosys Soft.
  - EntireX (SAG), Liquid Data (BEA)

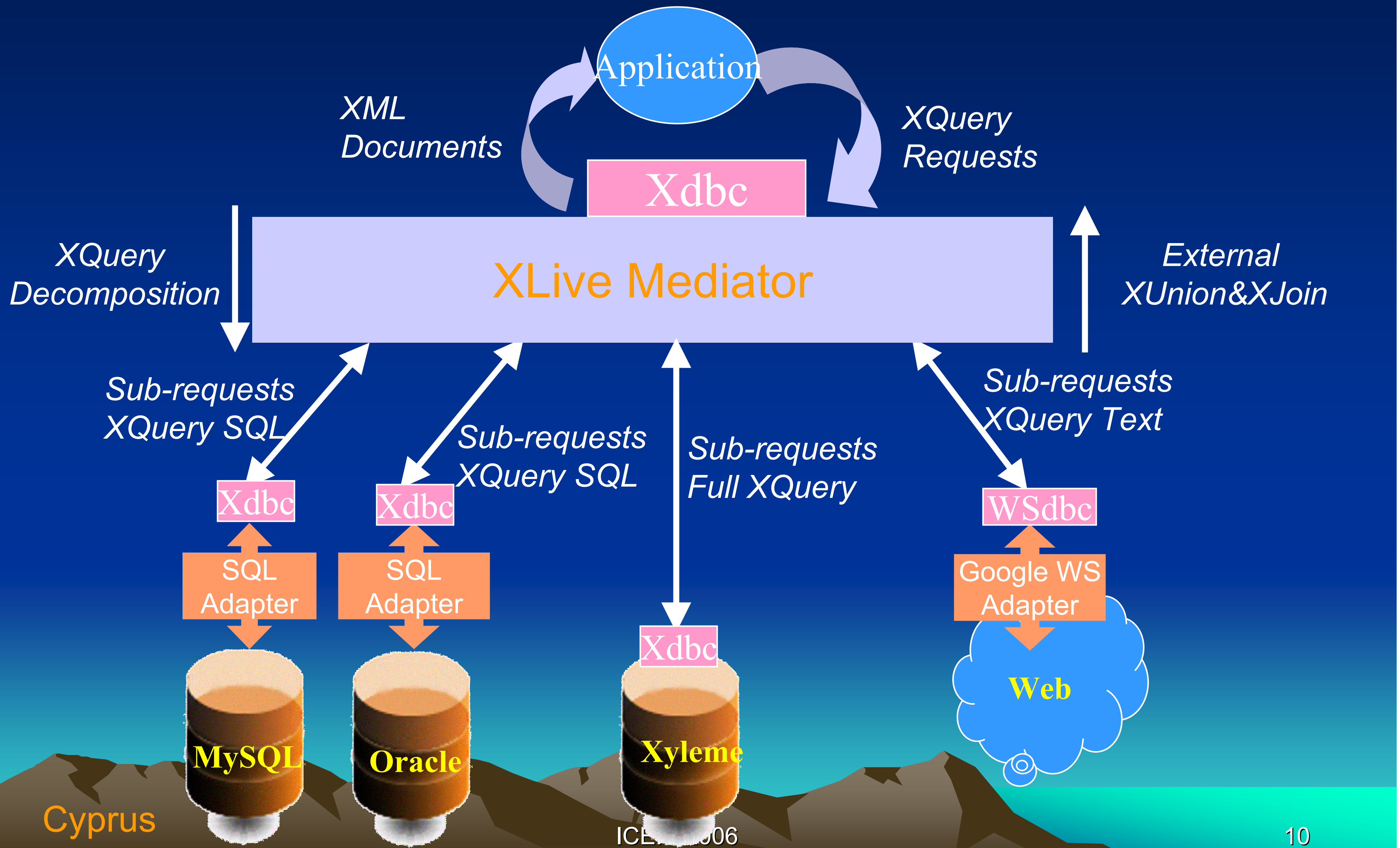
XML Mediation: Advantages

- Provide integrated access to heterogeneous sources through standard XML API (J2EE and Web Service)
- Retrieve and deliver up-to-date XML documents compound from multiple sources
- Assure transparency to source heterogeneity through a rich standard exchange model
- Ease the development of adapters for tree semi-structured data and text
- Provide rich meta-data to describe and localize data
- Increase availability of data sources through cache and concrete views

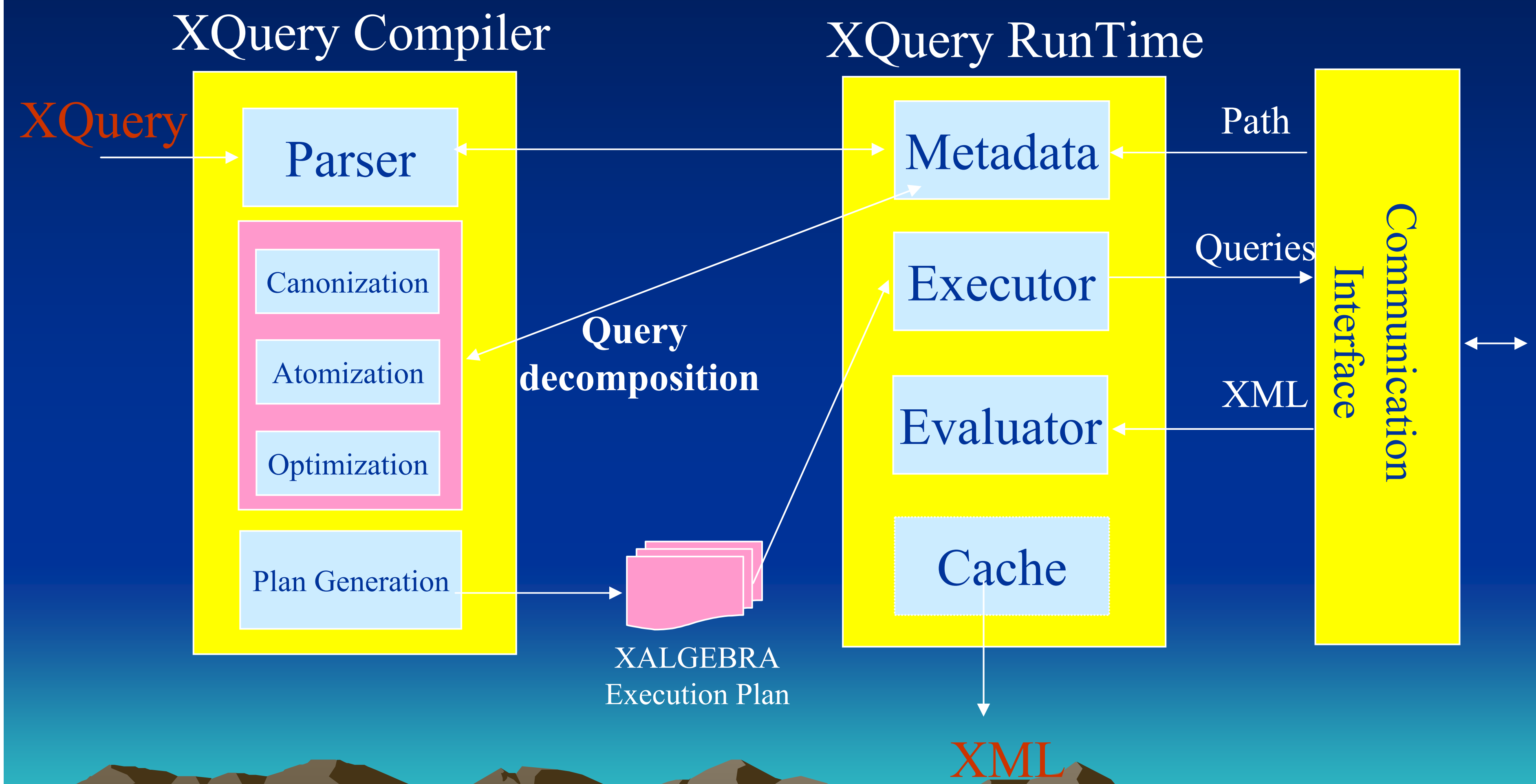
2. Xlive

- A mediator developed in EE projects at PRiSM
- Provide XQuery access to integrated XML views
  - Java XQuery API (XDBC)
  - Web Services API (WSDBC)
  - Each source is XQuery adapted (wrapped)
- Transparency to data localization and efficiency
  - Determine sources by schema element names
  - Parameterized query optimization (XAlgebra)
  - Text queries supported through semi-concrete views

Xlive : Overview



Xlive: Architecture



Extension: More Semantics

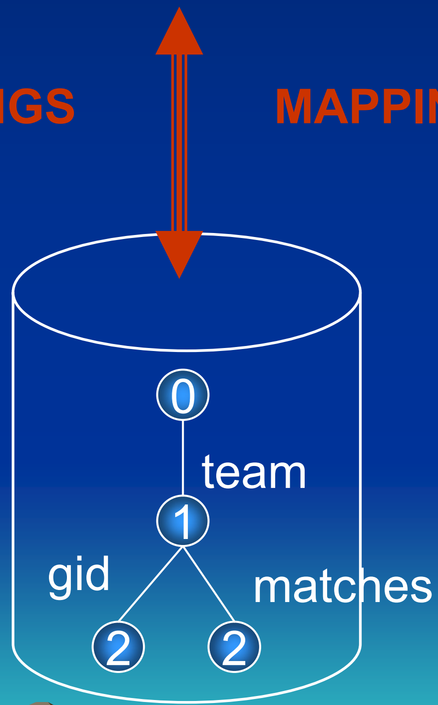
- Ontology supported integration
  - An ontology is a consensual and formal vocabulary to describe a specific domain
  - Class, relationships, attributes, instances, rules … → W3C OWL language
  - e.g. Football Worldcup Ontology
- XML schemas should be expressed in terms of one or more ontology
  - schema mapping could be written by hand
    - XQuery is a powerful tool to express mappings
  - schema mapping could be derived automatically
    - a good challenge ! description logic or datalog+ ?

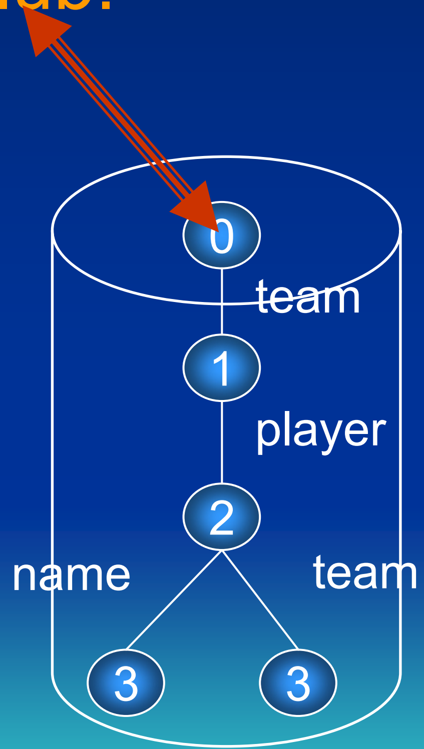
Example

- player∈team; player⇒name; team∈group; team ⇒*matches; group⇒gid; team⇔club.

MAPPINGS MAPPINGS

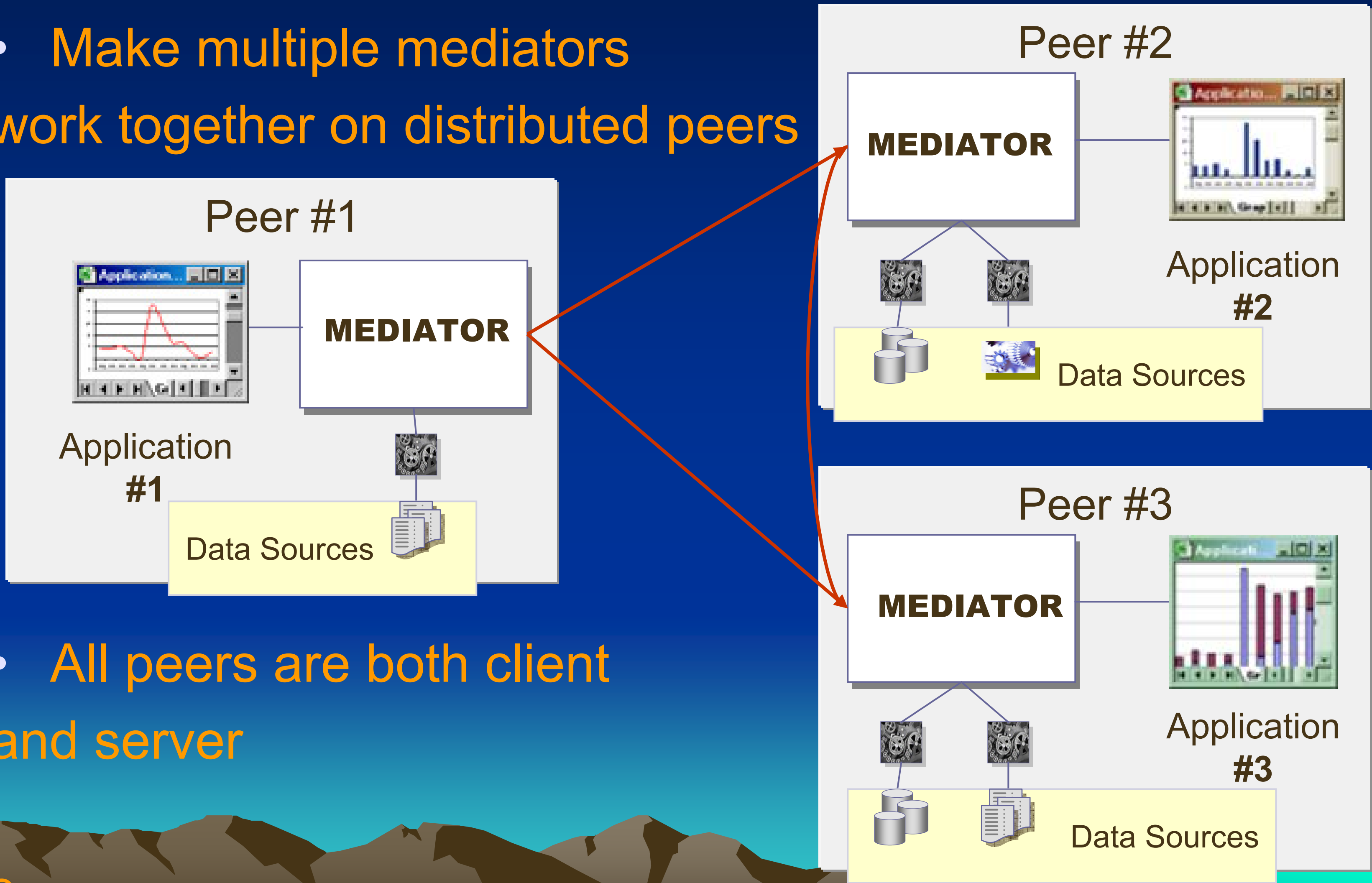
for $p in player*, $t in team*
where
$p/player/team=$t/group/team
and $t/group/id="F"
return
{$p/player/name}





3. P2P Mediation

- Make multiple mediators work together on distributed peers



- All peers are both client and server

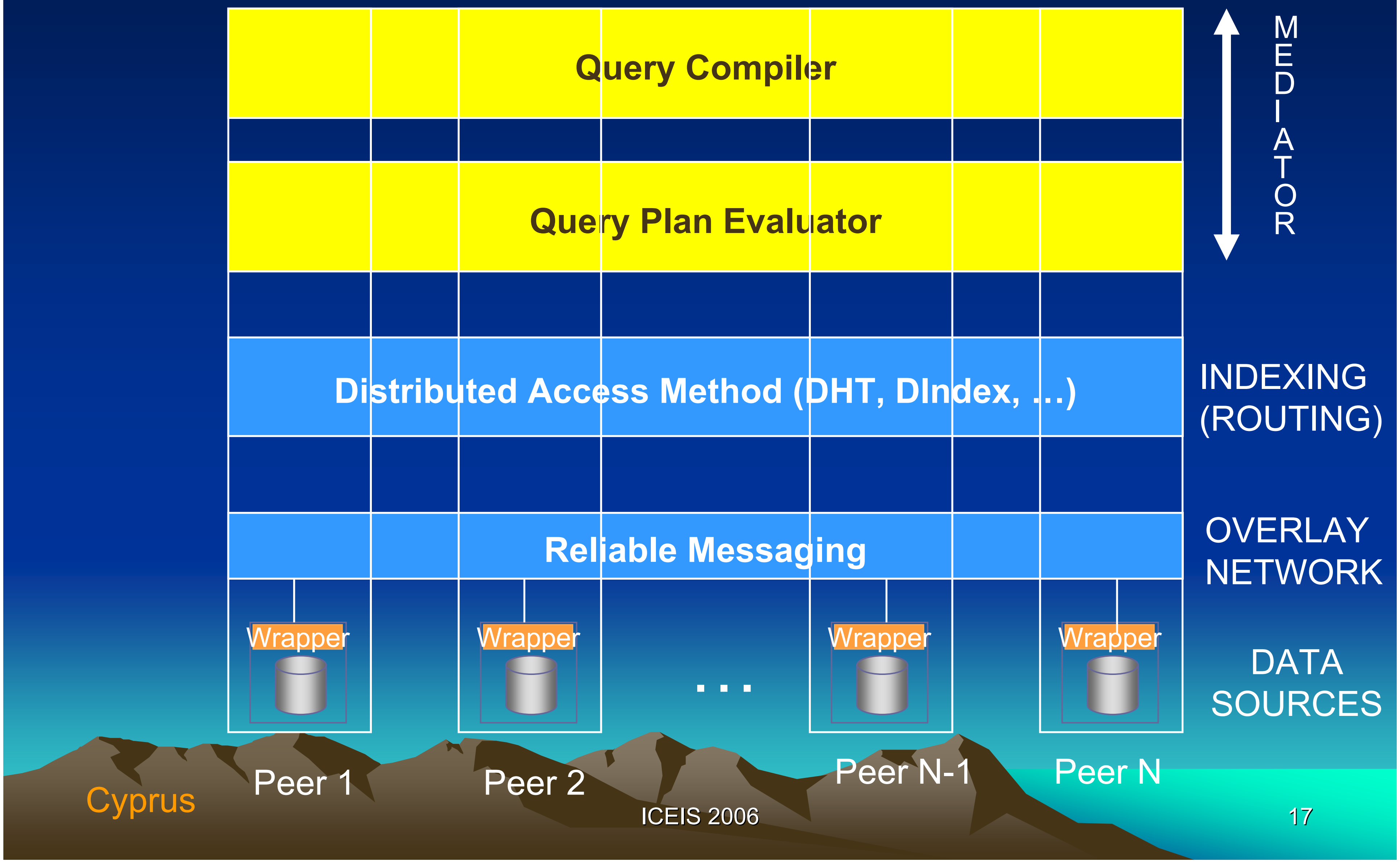
Some Advantages

- Scalability
  - Query can involve a large number of nodes
  - Several mediators can work in parallel
- Symmetry
  - Each node installs the same components (servent)
  - Should include facilitator, mediator, and adapter
- Multiviews
  - Each node can publish views of internal/external data
  - Semantic mappings can be achieved through multiple levels of views
- Openness
  - Node can dynamically connect and disconnect to the network

Some Problems

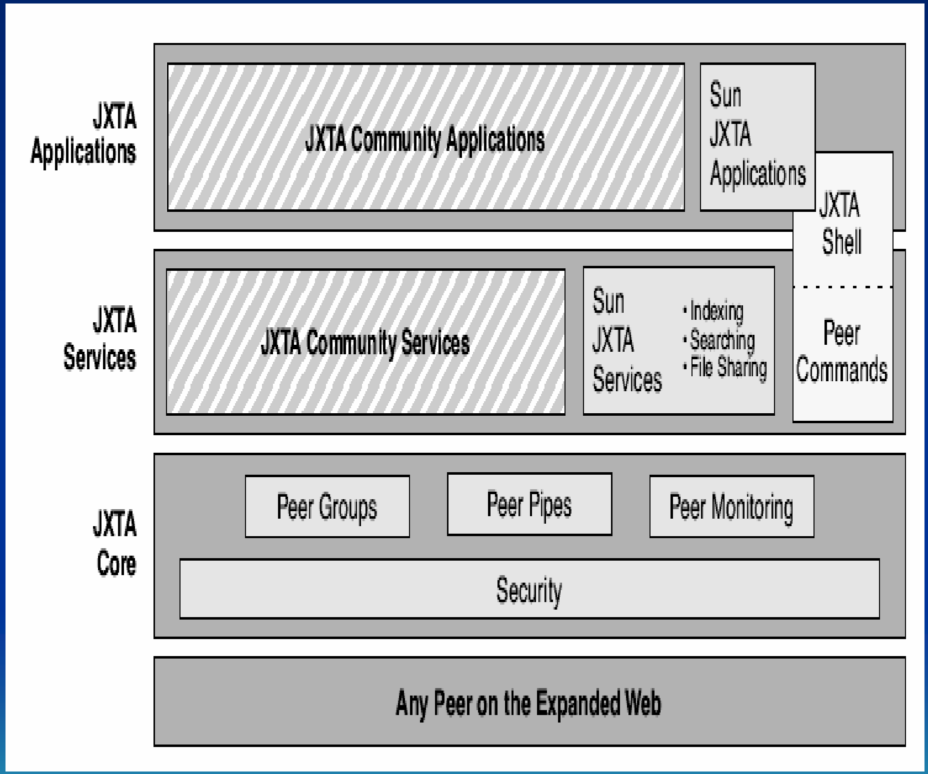
- Localization of relevant data sources
  - Nodes publishes XML views definition in a common language
  - Paths “indexed” to determine relevant views for a query
- Semantic translation and mappings
  - Each application bases collaboration on an ontology
  - Wrappers should map local ontology to the application ontology
- Parallel query processing
  - Localized data sources are highly distributed
  - Query plan should be optimally distributed on peers
- Fault-tolerance and security
  - …

P2P Mediation: Architecture



Access Methods

- Unstructured, structured, hybrid
  - CHORD = distributed H-table of keys and peers placed around a circle
  - CAN = partitions a N dimensional space into zones owned by peers
  - Pastry, Tapestry = based on hypercube topology
  - P-GRID = Binary tree auto-adaptable
  - …
- Determine the route of messages on the overlay network
- Incorporated in plate-forms
  - JXTA from Sun



Query Processing

- Source localization
  - Generation of localization request
  - Based on metadata (e.g. schema of views)
  - Using distributed indexes possibly built on DHTs
- Query evaluation
  - Centralized: collect the relevant source view fragments and compute results on client mediator
  - Distributed: migrate query (plans) from peer to peer with (reference to) selected data and compute results (e.g. joins) using distributed algorithms
  - Data should be as much as possible reduced locally

Structural Routing Projects

- XPeer (Sartiani, 2003)
  - Sharing of XML data with XQuery
  - Hybrid architecture with indexing super-peers and data peers
  - No schema mapping
  - Self-organizing XML P2P database system
- Oregon Univ. system (Papadimos, 2003)
  - P2P architecture for querying XML distributed sources
  - Queries are routed based on distributed catalogs
  - A query is processed by visiting relevant peers and replacing at each node part of the query plan by local XML data
- MediaPeer (Dragan, 2005)
  - Sharing of XML data with XQuery
  - Hybrid architecture with indexing super-peers (Patricia trie)
  - Limited schema mappings through wrapper views

Semantic Routing Projects (2)

- Piazza (Halevy, 2003)
  - A P2P infrastructure for sharing and mediating XML and RDF sources
  - XML peers export XML schemas describing local sources while RDF sources export OWL ontologies
  - Mappings between schemas (or ontologies) provided in XQuery
- SomeWhere (Adjiman, 2005)
  - Smilar to Piazza, but uses description logic to define mappings
  - Queries are routed according to the relevant mappings
- Kadope (Abiteboul, 2005)
  - A P2P architecture for sharing and mediating XML resources
  - Structural and semantic paths are indexed based on a DHT
  - Semantic model based on isA, part of, relatedTo relationships
  - Queries expressed directly over the XML types, semantics links

4. PathFinder: Objectives (1)

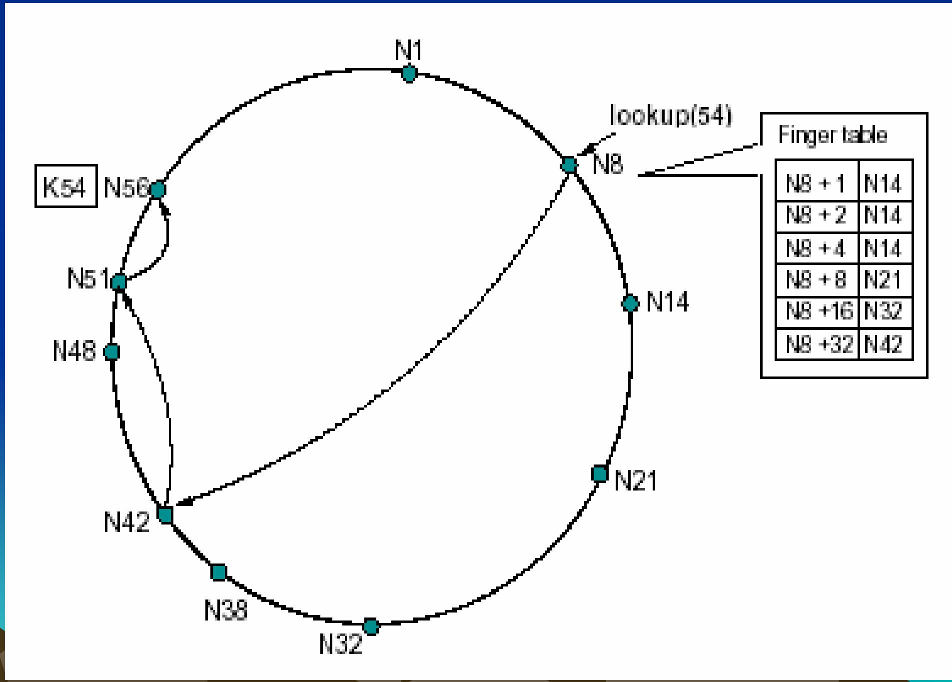
- Currently being prototyped and evaluated at PRiSM
  - P2P mediation system with a large number of servents
  - Distributed search engine based on XML/XQuery [text]
  - Based on XLive mediator deployed at each peer
  - Each peer publishes XML views (➔paths) of the local data
- A query is expressed on a client view
  - The system must efficiently localize the relevant source views
  - The query is processed using XLive on the relevant views
- Key ideas:
  - Use XML paths to publish and retrieve data source views
  - Use a DHT to index and localize relevant paths
  - Use a preserving order Hash-function for range queries

Selected Indexing Method

- Structured system based on a DHT
  - Decentralized, Self-organizing, Scalable
  - Fault tolerant
  - Guaranteed lookup complexity (≤ log N)
- Chord model:
  - Model adapted to our path indexing requirements
  - Keys are paths and contents are source addresses
  - The consistent H-function is replaced by an order preserving function and overflow management

Chord : Some Recalls

- Keys and peer-IDs are hashed to a ring (M bits)
- Store each key at first node hashed equal or above
- Skip-lists (Fingers) are used to accelerate search (<log N)
- Example: M=6 (N0 to N63), lookup(54) issued at N8



PathFinder Indexing Method

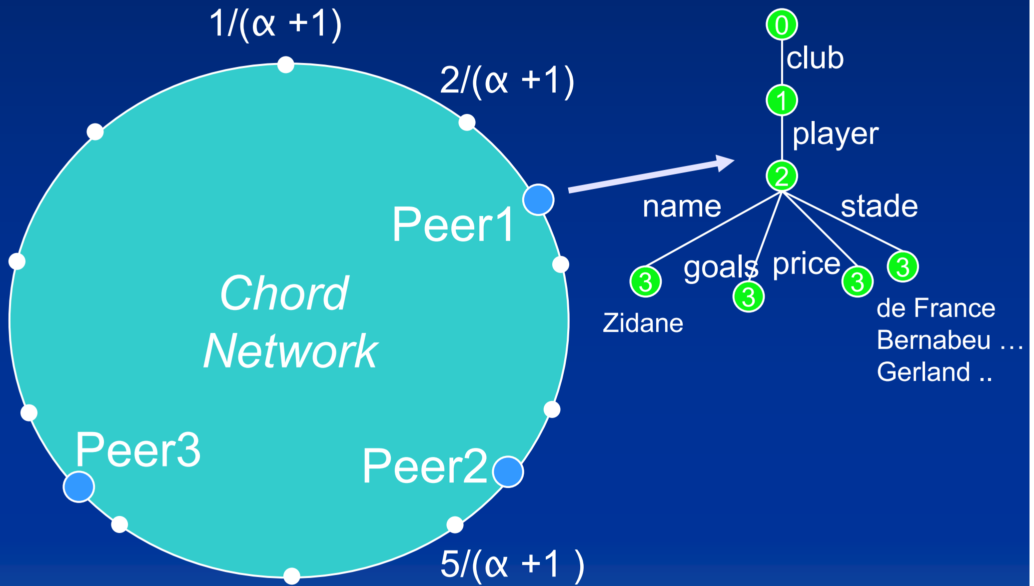
- Path clustering :
  - Adaptation of Chord indexing method:
    - use paths as keys
    - use a path preserving order hashing function
  - Paths with similar prefix are placed at the same peer (as much as possible)
  - Paths are mapped to identifiers in $0..2^m$
- What about consistent hashing ?
  - Overflow mechanism
  - Redistribution of sub-paths when a peer is overloaded

Path Hashing Function

- XML path mapping:
  - Each string can be mapped to a fractional number between 0 and 1 (Jagadish, 2000)
  - Let t1/t2/…/tn be a path; e.g. club/player/name
  - Hash each element ti to a numerical value:
    - Hash function h distributed to each peer
    - Hash function h must keep lexical order (for range query)
  - Example: H("club")=1 ; H("player")=3; H(name")=2
- Compute the path hash value in fractional numerical basis (α+1), i.e., giving more importance to prefix:
  - α = hashing domain
  - $H(path) = h(t1)/(\alpha+1)^1 + h(t2)/(\alpha+1)^2 + \ldots + h(tn)/(\alpha+1)^n$
  - Example:
    - hashing domain α = 20
    - $H(club/player/name)= 1/21^1 + 3/21^2 + 2/ 21^3$
- The selected node is the first greater or equal to 2^m x H(path)

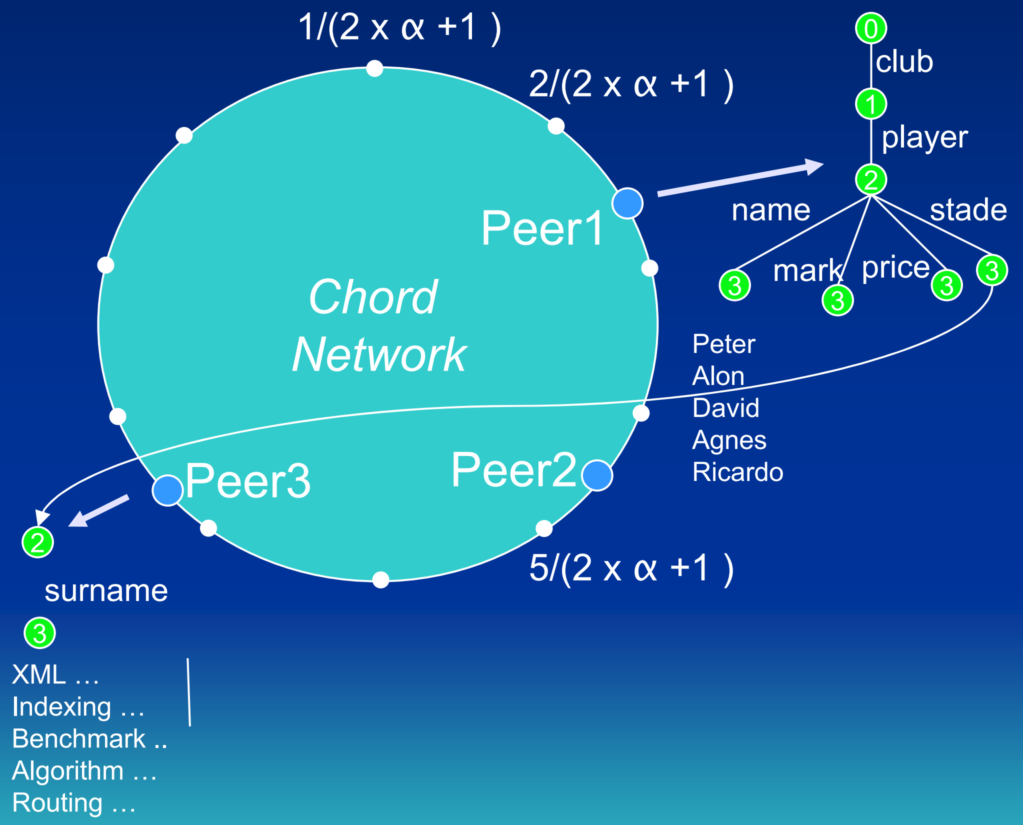
Illustration

- Based on the previous mapping method, XML paths are indexed in Chord
- XML paths with similar prefix are indexed by the same peer
  - e.g., Peer1 indexes all paths with prefix club
- Several elements have complex values
  - can be indexed locally
  - e.g., "goals", “stade”



Overflow distribution

- Path clustering → no consistent hashing:
  - The path load is not uniformly distributed to peers
- Overflow solution:
  - Re-index sub-paths from an overloaded peer
- Observations:
  - Keep maximal prefix at the same peer
  - Clustering to the same peer → cluster on different prefix
  - Uniform path distribution might not be reached BUT clustering maintained

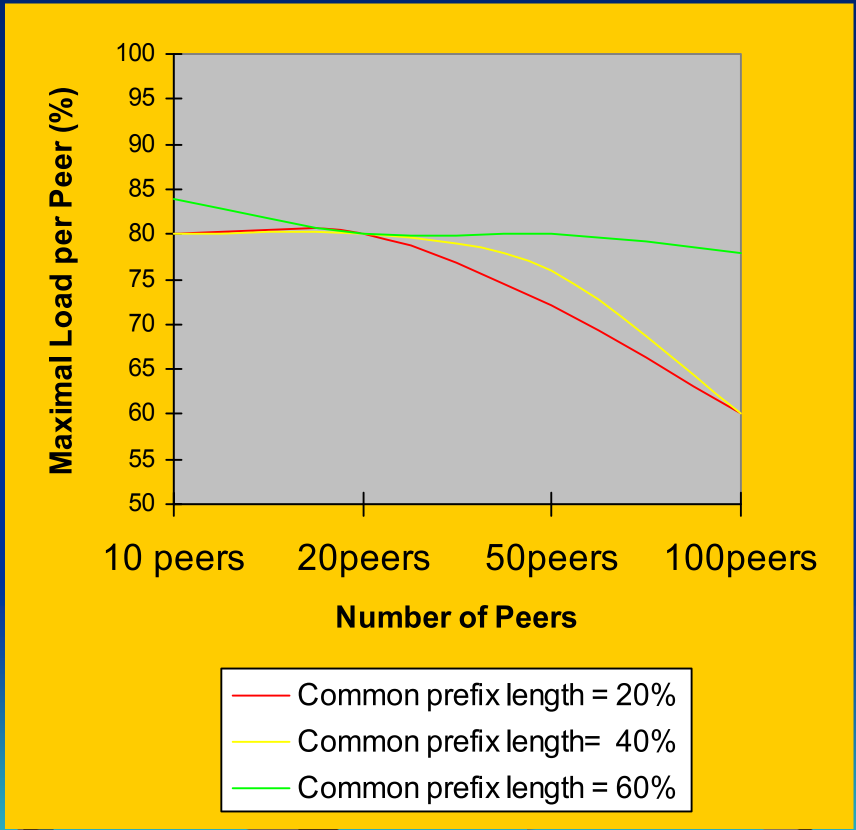


Query evaluation

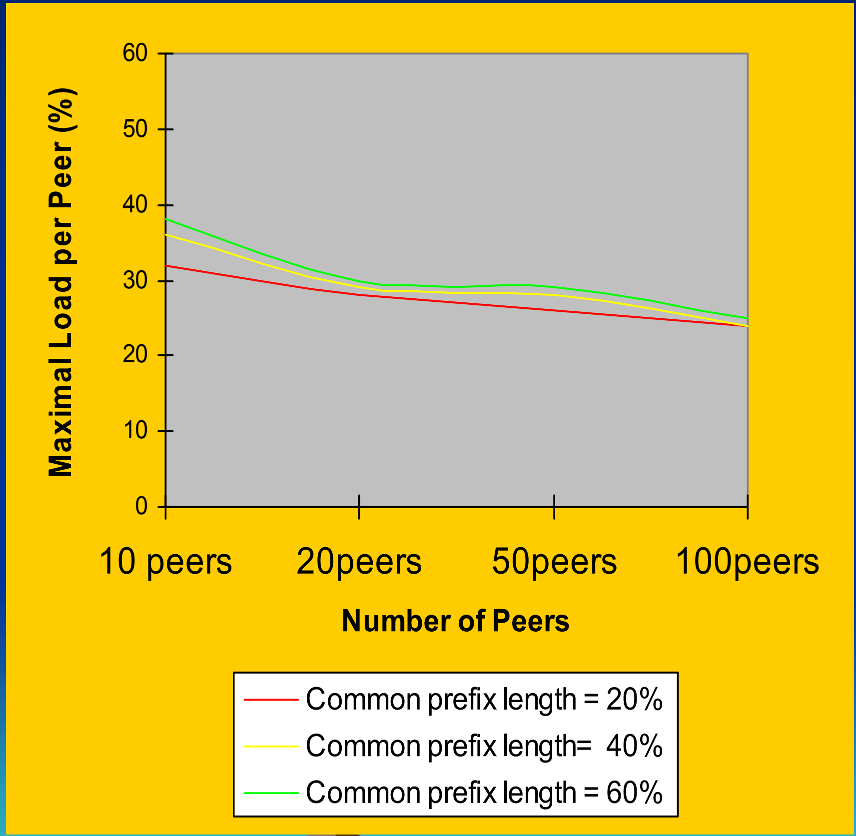
- Model efficient for twig-query (with common prefix in searched paths) :
    - Evaluate the common prefix
    - Starting from the peer indexing the common prefix, evaluate the rest of the sub-paths
- Example:
  - *for $p in collection("club")/player where $p/name = "Zidane" and contains ($p/stade,"France") return $p/price*
    - First route: /club/player –> Map to numerical value and route the whole query to the corresponding peer
    - Depending on the locally indexed values, evaluate the remaining sub-paths

Experimental Evaluation: Load Distribution

PathFinder

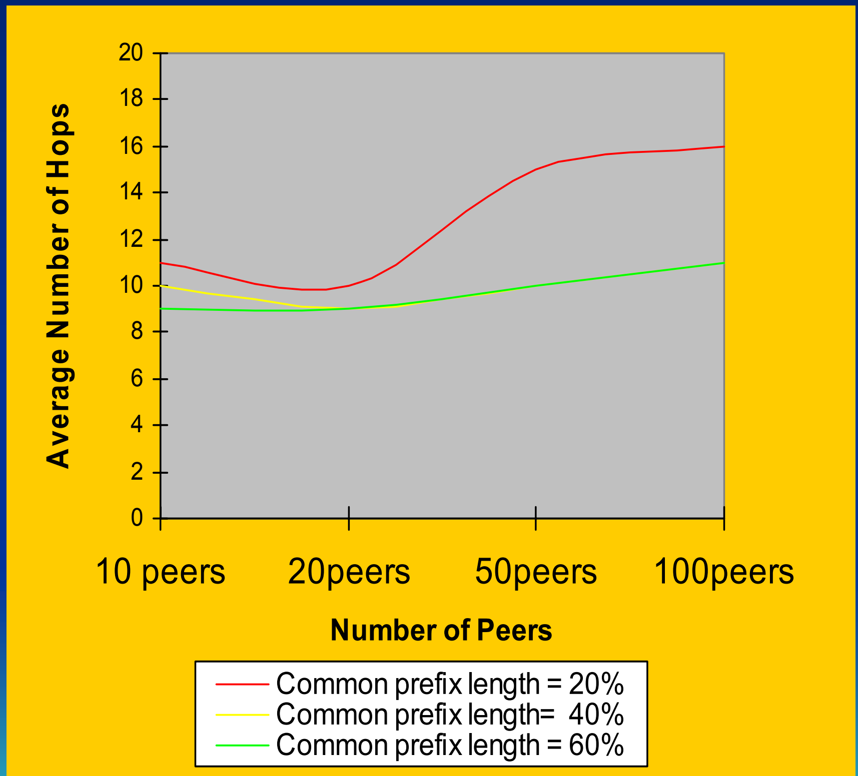


Chord

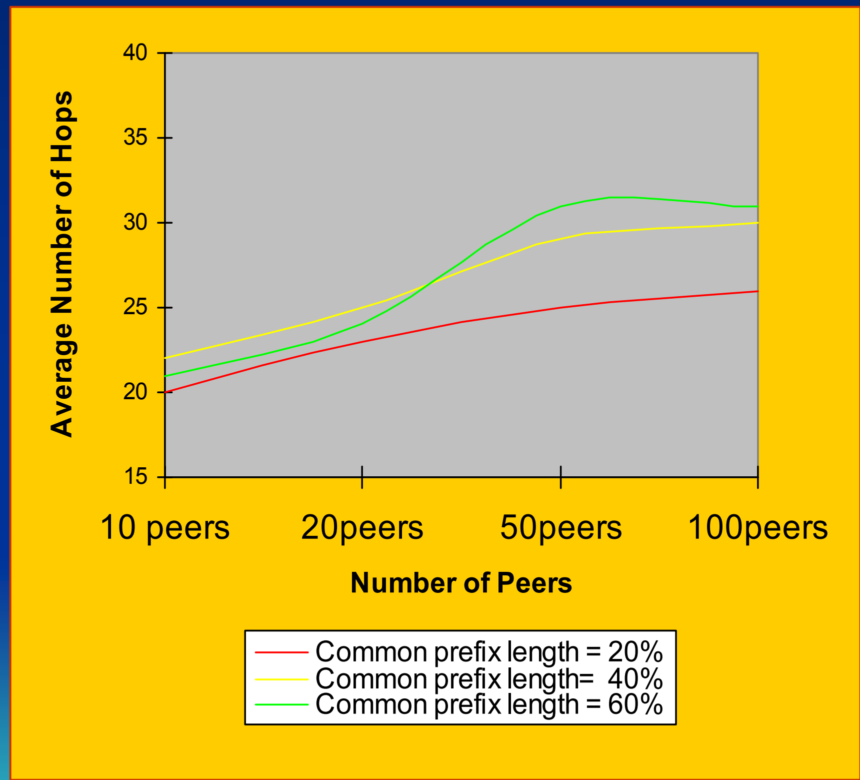


Experimental Evaluation: Query Routing Algorithm

PathFinder



Chord



5. Conclusion

- P2P mediation is a promising technology
  - Fast routing of queries on large networks
  - DHT on paths extendable to range queries
- Semantic integration of sources
  - Ontological mapping of schemas
  - Semantic XQuery, query enrichment
- Multiple access methods for XML paths
  - Have been proposed and more can be proposed
  - A comparative benchmark plate-form is required