Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Lehrstuhl für Rechnernetze und Telematik

Seminar: P2P Netzwerke

Topic: Semantic P2P Database, RDF

Patrick Tchakoute

29.07.2009

Overview

- Motivation
- RDF (Definition and example)
- RDFPEERS Architecture
- DHT(Chord)
- How to store?
- Query
- Evaluation
- References

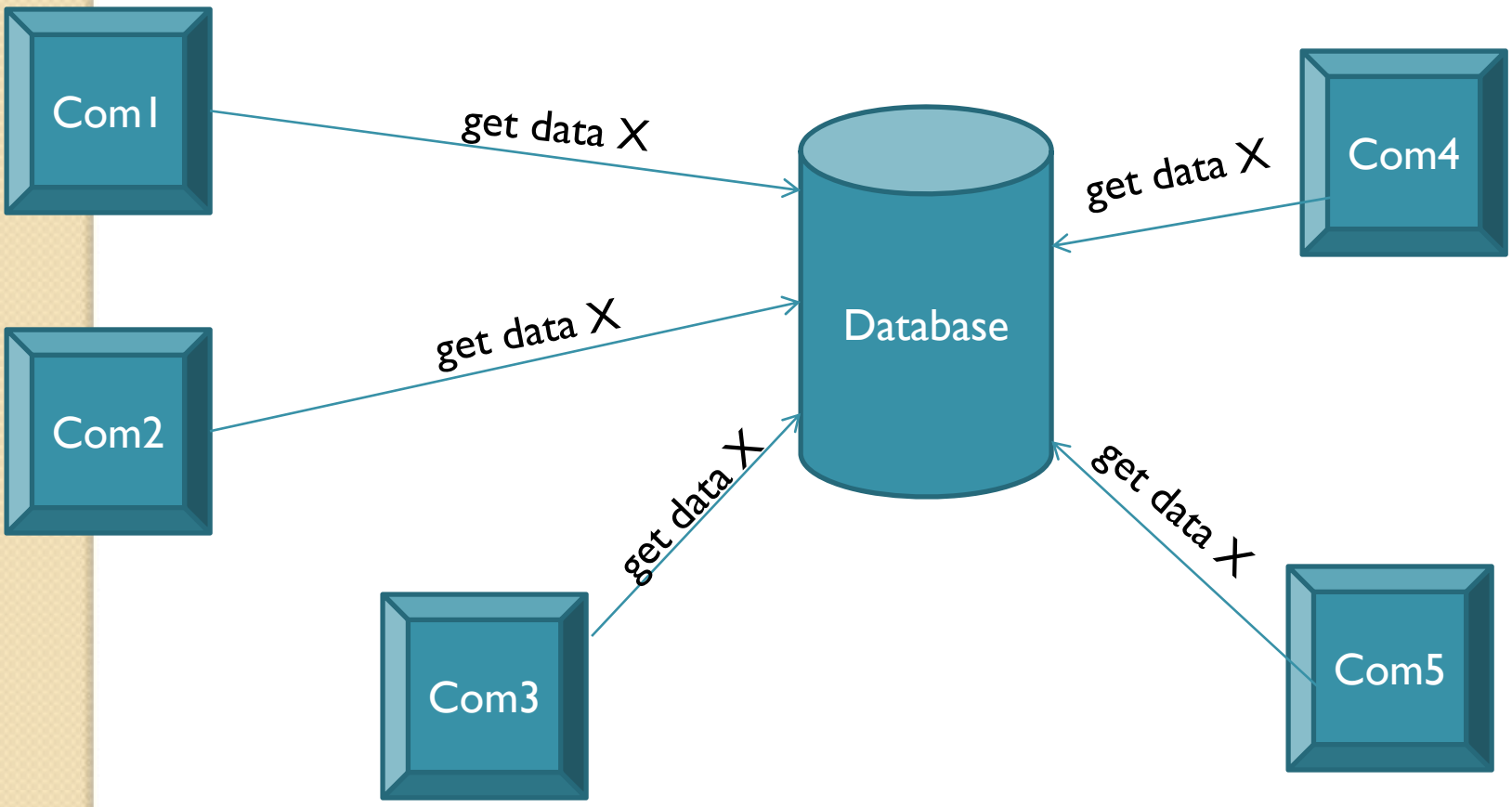
Motivation

- Can we guarantee to find data („query results“) ?
- NO

- What happen in a highly dynamic environment where many data join and leave at any time?
- Registration bottleneck

Motivation

- Overloaded system



Motivation

- No system failure by growing amounts of traffic

- Scalable and distributed RDF repository presented by M. Cai and M. Frank („RDFPeers“)

RDF
Definition

- RDF = Resource Description Framework
- RDF represents resources on the web
- Special form so that computer can understand

RDF Definition(2)

- RDF documents are composed of a set of RDF triples

<subject, predicate, object>

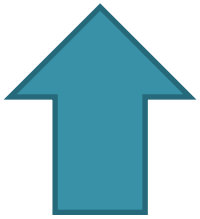
RDF Definition(3)

- Subject is either an URI(Uniform Resource Identifier) or a blank node
- Object is either a resource or a literal
- Resource is identified by URI
- Literals are either plain or typed
- Literals have the lexical form of a unicode string

RDF
Example

Information: „*rdfpeers creator is mincai*“

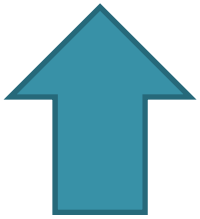
<info:rdfpeers> <dc:creator> <info:mincai>



resource object

RDF
Example(2)

- <info:mincai> <foaf:name> "Min Cai"

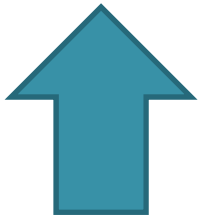


plain literal object

RDF Example(3)

Information: mincai is 28 years old

- <info:mincai> <foaf:age> "28"^^<xmls:integer>



typed literal object

RDFPEERS Architecture

- RDFPeers is organized into a multi-attribute addressable network(MAAN)

- Each node consists of five components:

1) the MAAN network layer
2) the RDF triple loader
3) the local RDF triple storage
4) The native query resolver
5) the RDQL-to-native-query translator

RDFPEERS Architecture(2)

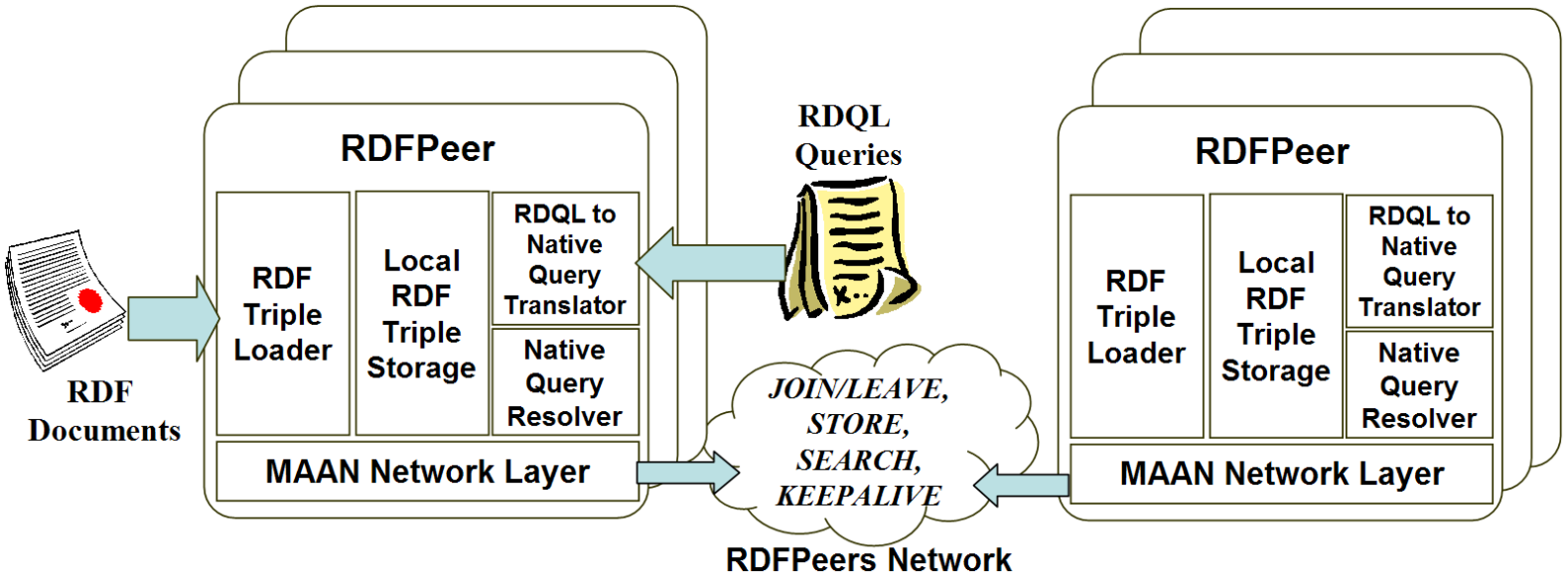
- MAAN protocol → 3 classes of messages
  - Topology maintenance
  - Storage
  - Search
- RDF triple loader:
  - read RDF document
  - parses it into RDF triples
  - Store triples into RDFPeers („STORE“ message)
- Local RDF triple storage:
  - Loacal storage of a RDPeer

RDFPEERS Architecture(2)

- Native query resolver:
  - Parses RDFPeers queries
  - Resolve queries („SEARCH“ message)

- RDQL-to-native-query translator :
  - Map high-level queries into native queries

RDFPEERS Architecture(3)



Source: RDFPeers: A scalable Distributed RDF Repository based on A Structured Peer-to-Peer Network; Min Cai and Martin Frank

DHT (Chord)

- System proposed by Ion Stoica (2001)
- Supports scalable <*key, object*> pairs registration
- Uses a one-dimensional circular identifier space (ring)
- Each node get a unique *m*-bit identifier (ID)
- Each node has a successor list and a finger table

How to store?

<info:rdfpeers> <dc:creator> <info:mincai> .
<info:mincai> <foaf:name> "Min Cai" .
<info:mincai> <foaf:age> "28"^^<xmls:integer>

- One of the three attribute values is designed as the destination of the routing
- Each triple is stored three times(once based on its subject, predicate, and object)
- Each triple will be stored at the successor node of the hash key of the value of the routing key attribute-value pair

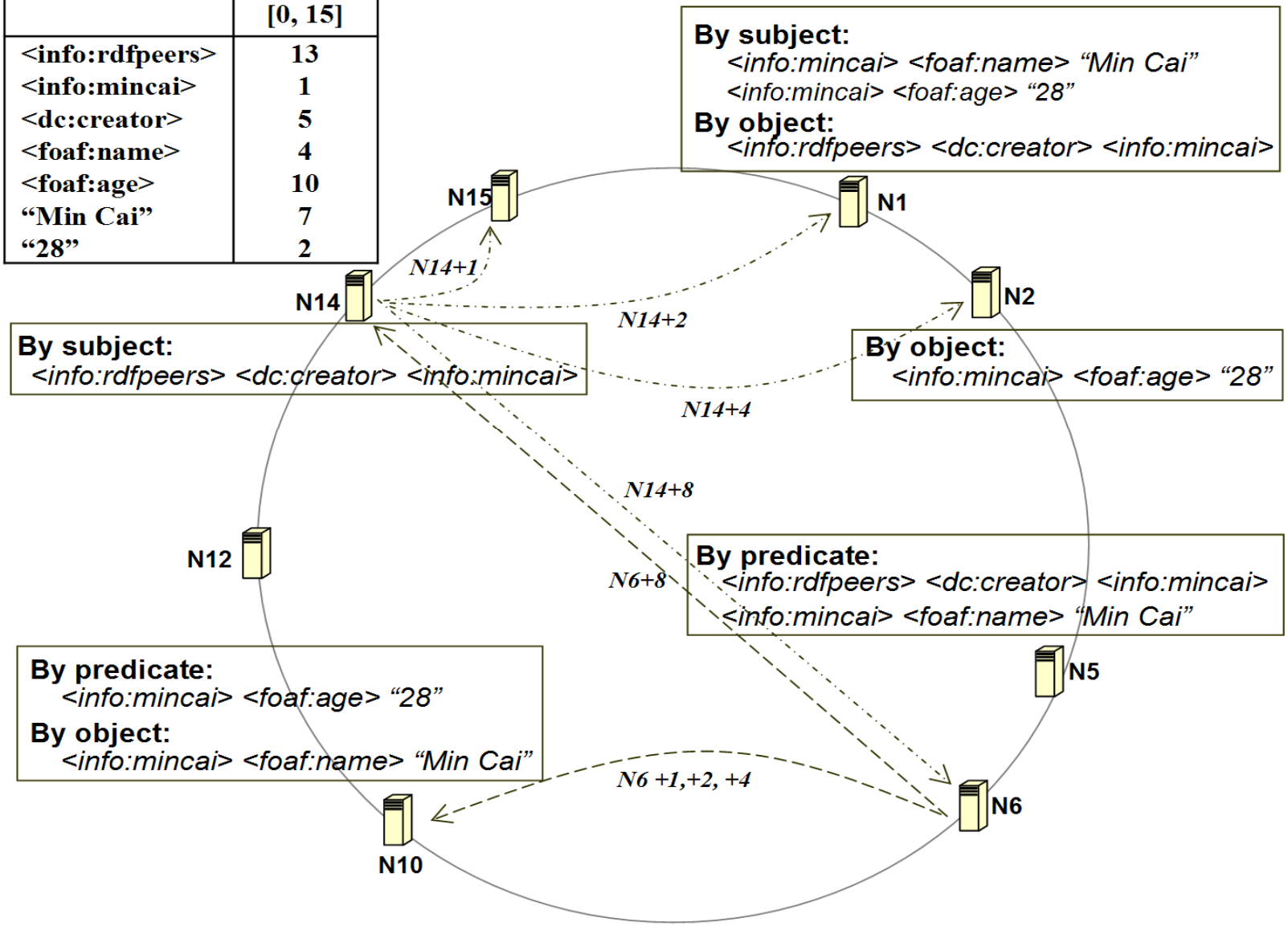
How to store? Example

- Store the first triple above by subject

STORE {key, {("subject", <info:rdfpeers>),
("predicate", <dc:creator>),
("object", <info:mincai>)}}
where key=SHA1Hash("<info:rdfpeers>")

- the first attribute-value pair *("subject", info:rdfpeers) is the* routing key pair
- *key is the SHA1 hash value of the subject* value.
- This triple will be stored at the node which is the successor node of *key*

URI / Literal	Hash Value in [0, 15]
<info:rdfpeers>	13
<info:mincai>	1
<dc:creator>	5
<foaf:name>	4
<foaf:age>	10
"Min Cai"	7
"28"	2



Source: RDFPeers: A scalable Distributed RDF Repository based on A Structured Peer-to-Peer Network; Min Cai and Martin Frank

Queries

- Atomic Triple Patterns
- Disjunctive and Range Queries
- Conjunctive Multi-Predicate Queries
- RDQL Queries (Jena Java RDF toolkit)
  - Inefficient for join queries

Queries

- **Atomic Triple Patterns**
  - triple pattern in which the subject, predicate, or object can each either be a variable or an exact value
  - eight resulting possible queries

No.	Query Pattern	Cost	Query Semantics
Q1	(?s, ?p, ?o)	O(N)	find all possible triples
Q2	(?s, ?p, o_i)	logN	given object o_i of any predicate, find the subjects and predicates of matched triples
Q3	(?s, p_i, ?o)	logN	given predicate p_i, find the subjects and objects of the triples having this predicate
Q4	(?s, p_i, o_i)	logN	...
Q5	(s_i, ?p, ?o)	logN	...
Q6	(s_i, ?p, o_i)	logN	...
Q7	(s_i, p_i, ?o)	logN	...
Q8	(s_i, p_i, o_i)	logN	...

Source: RDFPeers: A scalable Distributed RDF Repository based on A Structured Peer-to-Peer Network; Min Cai and Martin Frank

Queries

- **Disjunctive and Range Queries**

- Domain of the variables is limited
  - Atomic queries with constraints

- Constraint
  - OrExpression
  - an interval for a numeric value

Queries

- **Disjunctive and Range Queries(2)**

  No: Q9

- (a) (?s, dc:creator, ?c) AND (?c="Tom" || ?c="John")
- (b) (?s, foaf:age, ?age) AND (?age > 10 && ?age < 20)

Queries

- **Conjunctive Multi-Predicate Queries**

  - Conjunction of atomic queries patterns or disjunctive range queries for the same subject variable.

Queries

- **Conjunctive Multi-Predicate Queries**

Example of a subject variable with a list of restricting *predicate, object or predicate, object-range pairs*.

No: Q10

- (?x, <rdf:type>, <foaf:Person>)
  (?x, <foaf:name>, "John")
  (?x, <foaf:age>, ?age) AND ?age > 35

Queries

- RDQL Queries
- Query language for RDF
- (1) SELECT ?x

  WHERE (?x, <vcard:FN>, "John Smith")

  → Q4

- (2) SELECT ?x, ?fname

  WHERE (?x, <vcard:FN>, ?fname)

  → Q3

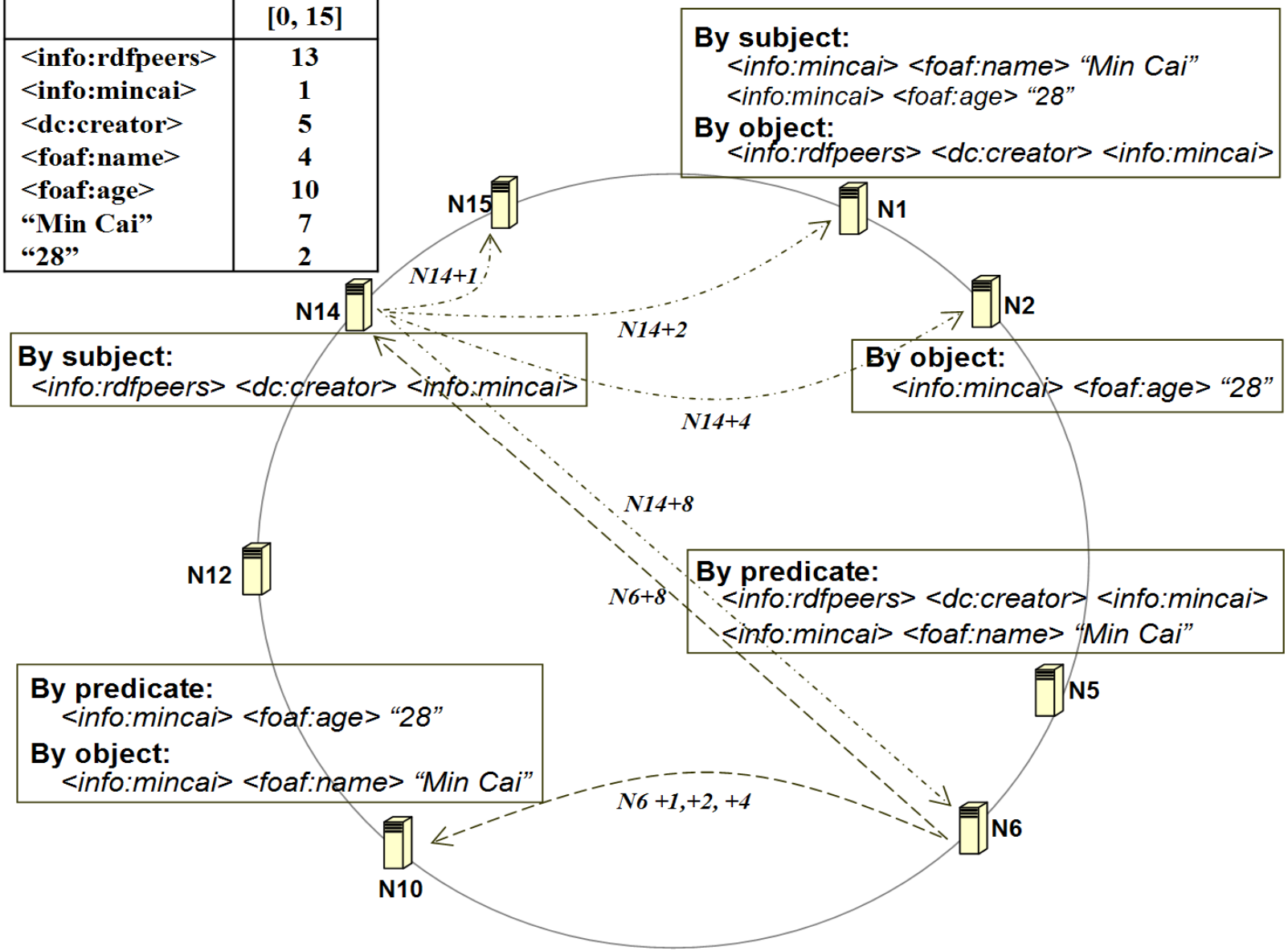
Queries

- RDQL Queries

- (3) SELECT ?givenName
  WHERE (?y, <vcard:Family>, "Smith"),
  (?y, <vcard:Given>, ?givenName)

  → Q9 with Q3

URI / Literal	Hash Value in [0, 15]
<info:rdfpeers>	13
<info:mincai>	1
<dc:creator>	5
<foaf:name>	4
<foaf:age>	10
"Min Cai"	7
"28"	2



Source: RDFPeers: A scalable Distributed RDF Repository based on A Structured Peer-to-Peer Network; Min Cai and Martin Frank

Evaluation

- Implementation in Java
- Performance measurement on a real-world network (128 nodes)
- Number of neighbors at each node increases logarithmically with the network size
  - → node state in MAAN scales well to a large number of nodes
- For exact-match queries, the number of routing hops in the worst case is *O(logN)* and the average routing hops is *(logN)/2*

References

- M. Cai, M. Frank.
  RDFPeers: A Scalable Distributed RDF Repository based on A Structured Peer-to-Peer Network

- M. Cai, M. Frank, J. Chen, and P. Szekely.
  MAAN: A multi-attribute addressable network for grid information services. In *4th Int'l Workshop on Grid Computing, 2003*.

- Peter Mahlmann and Christian Schindelhauer. Peer-to-Peer Netzwerke, pages 81–88.
  ISBN: 978-3-540-33991-5. Springer, 2007.

Thank you for listenning

Questions?