Using Peer-to-Peer Dynamic Querying in Grid Information Services

Domenico Talia and Paolo Trunfio

DEIS - University of Calabria

HPC 2008

July 2, 2008 - Cetraro, Italy

CoreGRID

Using P2P for Large-scale Grid Information Services

- **Information services** are key components in Grid systems to enable effective **discovery** and usage of distributed resources and services.
- Today Grid information services are based on hierarchical solutions, which can suffer limited scalability and reliability as the size of the system increases (e.g. large-scale Grids, massive clouds and pervasive service oriented systems).
- P2P architectures have been proposed as an alternative to hierarchical solutions to implement more scalable and reliable Grid information services:
  - Unstructured P2P systems
  - Structured P2P systems

Objective of this work (1/2)

- Distributed Hash Tables (**DHT**s) allow scalable resource discovery in structured P2P networks but do not support some types of queries (e.g., regular expressions) and cannot index dynamic information.

- On the other hand, supporting **arbitrary queries** in Grids is very important: In many scenarios resources must be dynamically located and selected on the basis of complex criteria or semantic features.

- We designed a way to support arbitrary queries in structured P2P networks by **implementing unstructured search techniques on top of DHT-based overlays**.

Objective of this work (2/2)

- We followed this approach by designing **DQ-DHT**: an algorithm that combines the *dynamic querying* (**DQ**) strategy used in unstructured P2P systems, with an efficient algorithm for *broadcast over a DHT*.
- The aim of **DQ-DHT** is two-fold:
  - allowing **arbitrary queries** in structured networks
  - providing **dynamic adaptation** of the search according to the popularity of the resource to be located.
- We evaluated **DQ-DHT** through simulations and experimented a prototype on a Grid testbed (Grid'5000).

DQ in unstructured networks

- **Goal**: controlling the query propagation on the basis of the desired number of results and the popularity of the resource to be located.
  - **Probe phase**: The search initiator sends the query towards a few neighbors with a small TTL to estimate the popularity of the resource.
  - If the desired number of results is not reached, an iterative process takes place to contact a new set of nodes.
  - The process stops when the desired number of results is received, or all neighbors have been already queried.

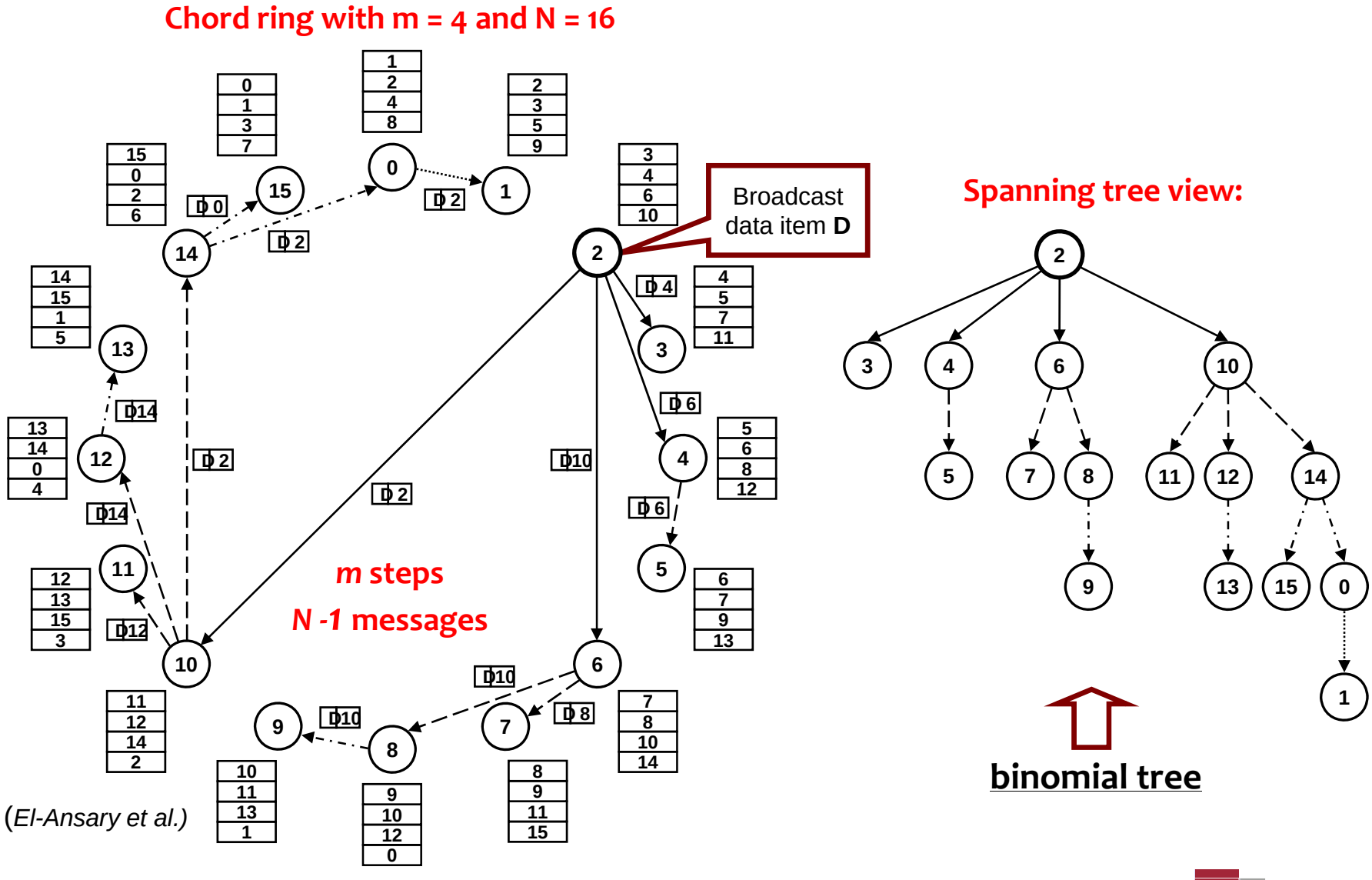
DQ in structured networks

- Similarly to DQ, **DQ-DHT dynamically adapts the search extent on the basis of the desired number of results** and the popularity of the resource to be located.

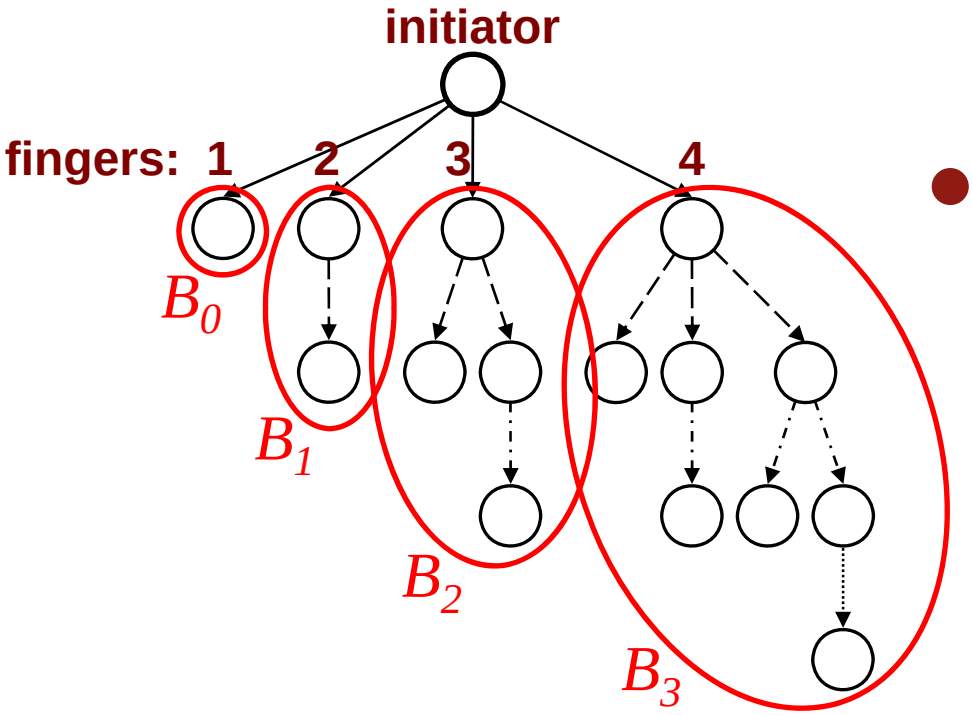
- Differently from DQ, **DQ-DHT exploits the structural constraints of the DHT to avoid message duplications.**

- DQ-DHT is based on an algorithm for broadcast over a Chord DHT (El-Ansary et al.) that allows to distribute a data item to all nodes in the network with N - 1 messages in $O(\log_2 N)$ steps.

Broadcast over a Chord DHT



Properties of the broadcast spanning tree



- Binomial tree B_i:
  - # nodes in $B_i = 2^i$
  - Depth of $B_i = i$
  - # nodes at level l in $B_i = \binom{i}{l}$

- Given the binomial trees properties, DQ-DHT can estimate with good approximation the number of nodes at the different levels of the subtrees associated to each finger (neighbor) of the querying node.

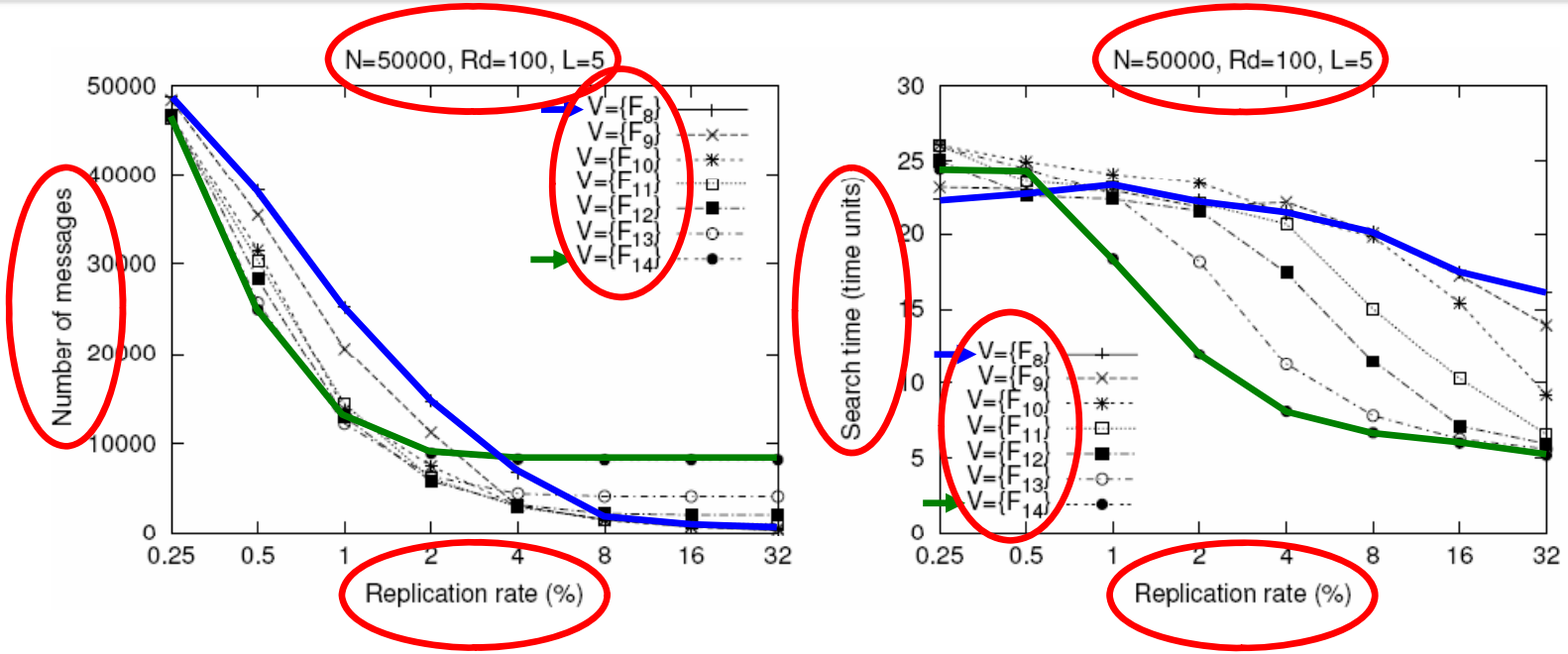
DQ-DHT algorithm

- **Node initiating a search:**
  1. Sends the query to a given subset of its fingers
     - waits for a given amount of time
  2. While (# results < # desired_results) and (more fingers to contact):
     1. calculates the item popularity
     2. chooses a new subset of fingers to contact
     3. sends the query to those fingers
     4. waits for a given amount of time.
- **Node receiving a query:**
  1. Processes the query locally against data it stores
     - in case of match it replies to the query initiator
  2. Forwards the query to the portion of the spanning tree it is responsible for, using the standard broadcast algorithm.

Performance analysis: simulation settings

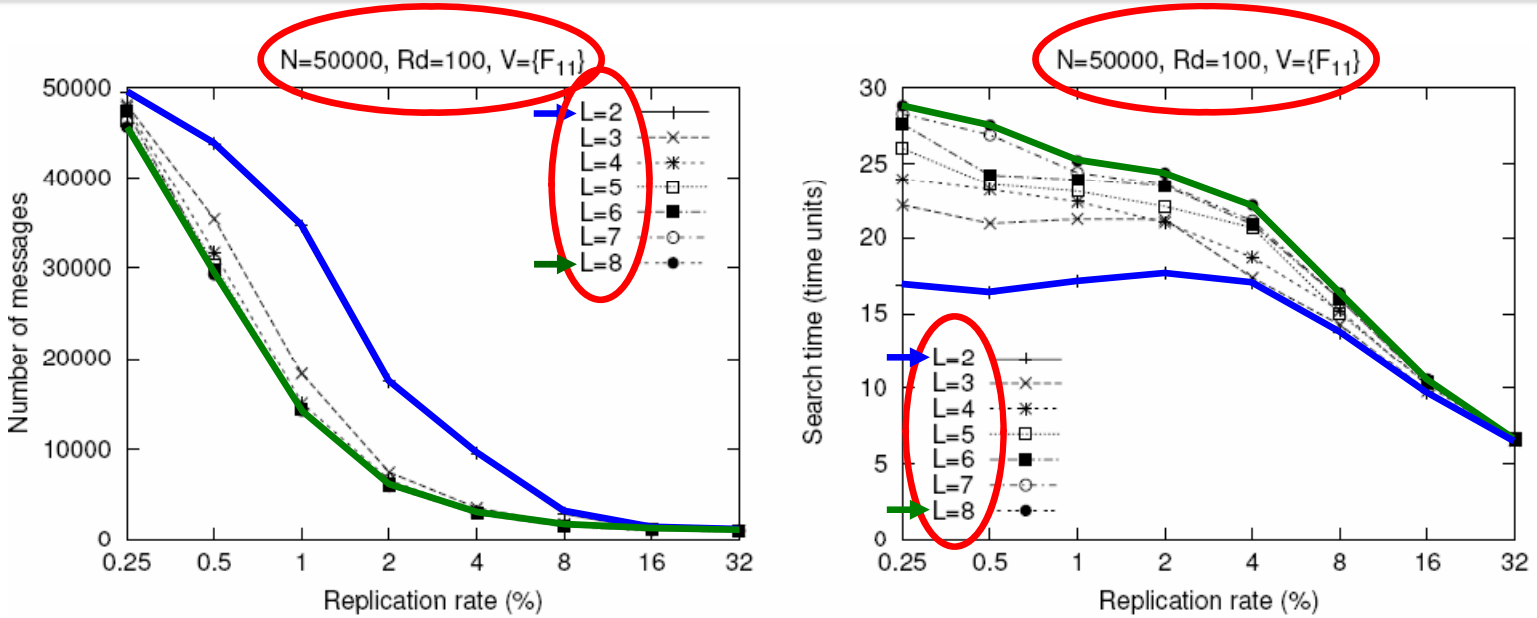
- **Network parameters:**
  - Identifier space: 32 bit
  - No. of nodes (***N***) = 50000
  - Resource replication rate (***r***): 0.25% to 32%
- **Algorithm parameters:**
  - Set of fingers to visit during the probe query (***V***)
  - No. of levels from which to wait a response during the probe phase, before to evaluate the resource popularity (***L***): 2 to 8
- **Query parameter:**
  - Desired number of results (R_d): 100
- **Performance metrics:**
  - Number of messages
  - Search time to obtain R_d results

DQ-DHT: choosing the value of *V*



- As expected, number of messages and search time decrease as ***r*** increases, for any value of ***V***.
- With low values of ***r*** (rare resources) it is convenient to start the search by contacting a finger rooting a large spanning tree (e.g., $V=\{F_{14}\}$), which leads to lower search times.
- When the value of ***r*** is unknown, it is convenient to start the search with an intermediate finger (e.g., $V=\{F_{11}\}$), which produces a good balance between search time and number of messages.

DQ-DHT: choosing the value of *L*

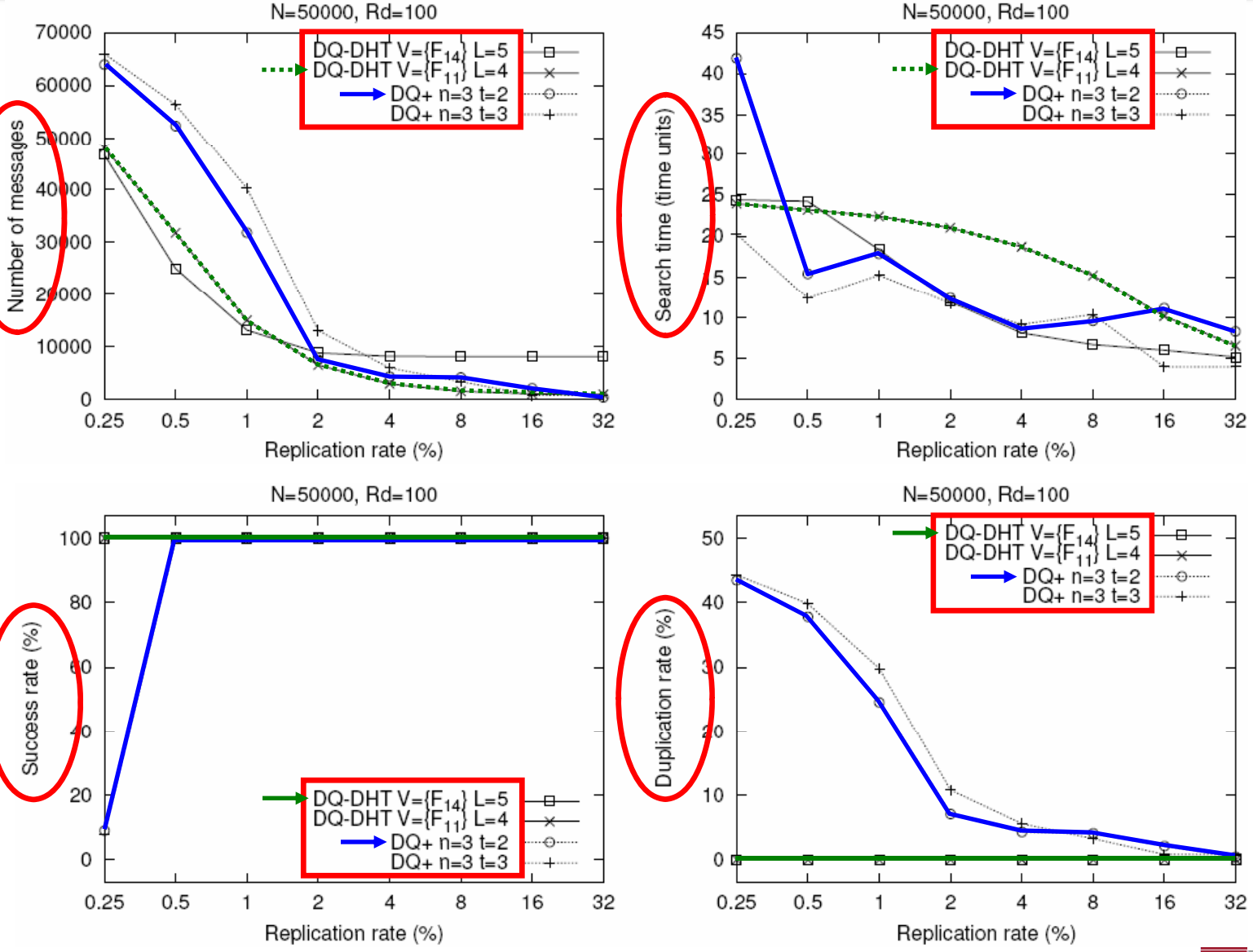


- Lower values of ***L*** generate lower search times, since the waiting time after the probe phase is proportional to ***L***.
- On the other hand, if ***L*** is too low, the resource popularity cannot be estimated in accurate way and a large number of messages will be sent.
- In general, intermediate values of ***L*** (e.g., L = 4 for $V=\{F_{11}\}$) produce the best compromise between number of messages and search time.

DQ-DHT: comparison with DQ (1/2)

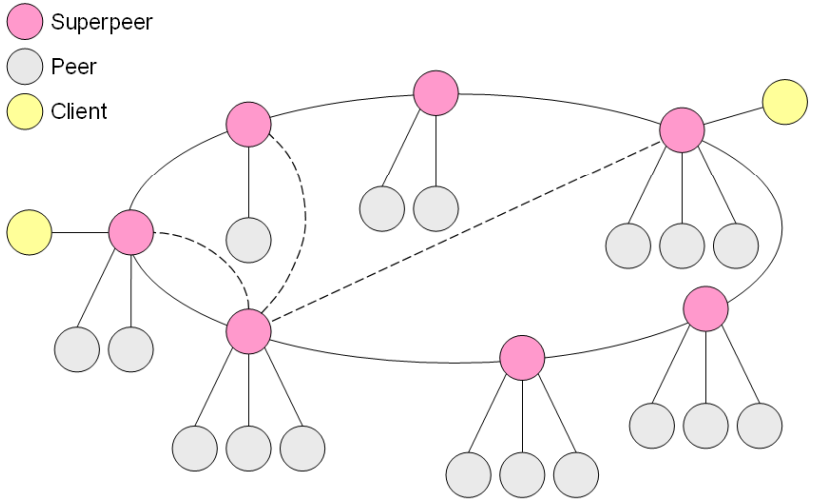
- **Performance metrics:**
  - Number of messages
  - Search time
  - Success rate
  - Duplication rate
- **DQ-DHT parameters:**
  - ***V*** and ***L***:
    1. $\boldsymbol{V} = \{F_{14}\}$ and $\boldsymbol{L} = 5$
    2. $\boldsymbol{V} = \{F_{11}\}$ and $\boldsymbol{L} = 4$
- **DQ parameters:**
  - ***n*** (# neighbors contacted during the probe phase) and ***t*** (time-to-live used to propagate the query during the probe phase):
    1. $\boldsymbol{n} = 3$ and $\boldsymbol{t} = 2$
    2. $\boldsymbol{n} = 3$ and $\boldsymbol{t} = 3$

DQ-DHT: comparison with DQ (2/2)



Using DQ-DHT in a real Grid scenario (1/2)

- Evaluating the use of DQ-DHT in a Grid scenario *:
  - Superpeer architecture:
    - DQ-DHT used for distributing queries among Superpeers
    - Client-Server communication among Peers and Superpeers

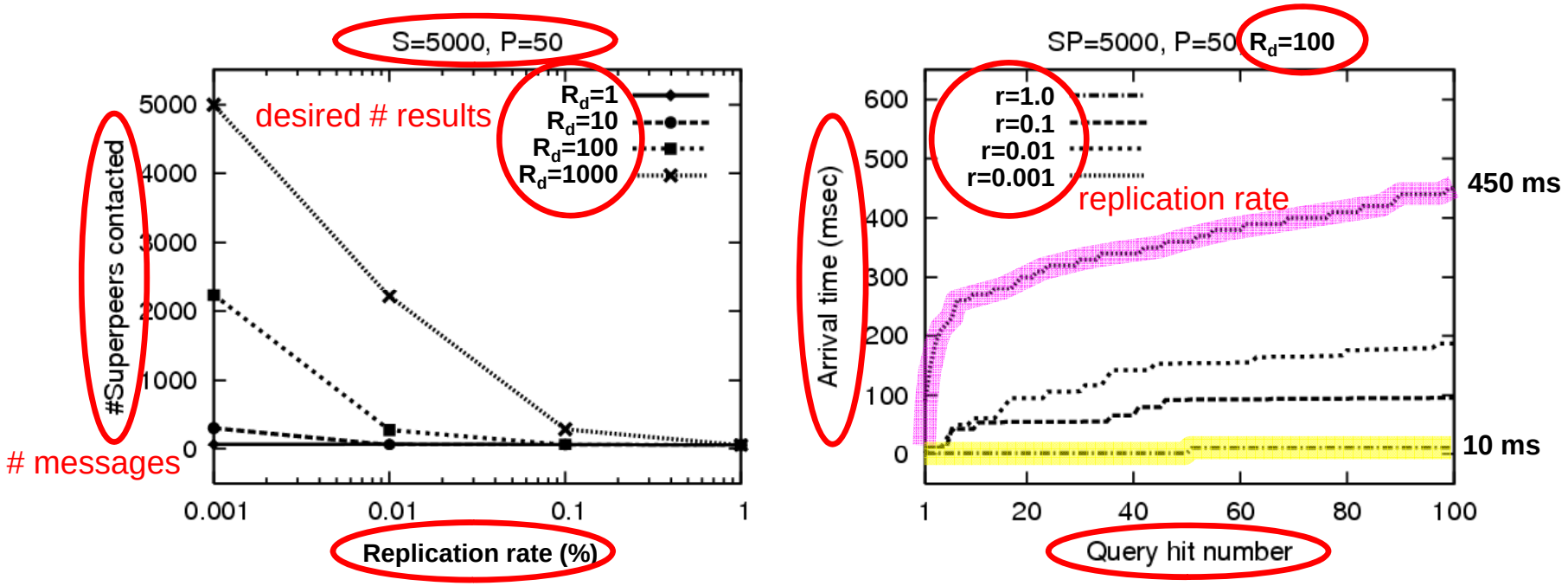


  - Implemented in Java using Open Chord

* H. Papadakis, P. Trunfio, D. Talia, P. Fragopoulou. "Design and Implementation of a Hybrid P2P-based Grid Resource Discovery System". In: Making Grids Work, Springer, 2008.

Using DQ-DHT in a real Grid scenario (2/2)

- Preliminary performance results on Grid'5000:
  - over 400 CPUs across 4 sites
  - 5000 Superpeers
  - 50 Peers per Superpeer



Concluding remarks

- In a pervasive scenario such as that of the **Internet of services** and the **Internet of things**, scalable information services are vital.
- Centralized or hierarchical approaches can be a bottleneck in such scenarios.
- The same occurs in large-scale dynamic distributed systems like **Grids**, **Clouds** and **P2P** systems: efficient service and/or resource discovery requires a scalable information service.
- Structured P2P models can provide effective solutions, but they must be flexible in handling **dynamic service discovery**.

Concluding remarks

- Combining structured P2P networks with unstructured P2P search techniques is an effective way to **support arbitrary queries** in distributed systems like Grids.

- We experimented the use of DQ-DHT as basic querying algorithm for the prototype of a Grid information service.

- Simulation and experimental results (on the Grid'5000 testbed) show the effectiveness of the approach.

- As future work we will study how to exploit the DQ-DHT approach to support **semantic service discovery** in emerging distributed environments like massive Cloud systems.

Questions?

Thank you