

A Lightweight Inter-node Operation for UDDI Cloud

Roberto Podesta'

ERCIM research fellow at INRIA,
roberto.podesta@inria.fr

Introduction

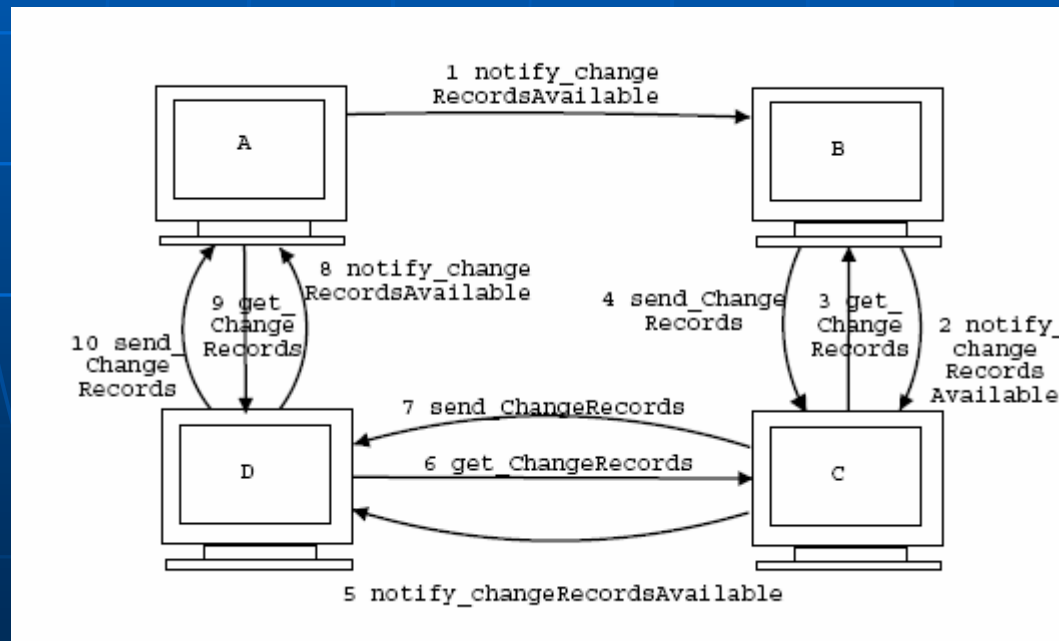
- UDDI federations (or clouds) disappeared with the end of UDDI Business Registry (UBR) project
- SAP, HP, IBM, Microsoft
- Why?
 - Natural end of the project (official explanation)
 - Lost of appealing for UDDI clouds
 - Why?
 - federation mechanism not efficient, not scalable, with huge overhead

UDDI Inter-node operation: Replication APIs

- `notify_ChangeRecordsAvailable(...)`:
 - advertisement of new available records to another registry
 - performed by the node which starts the update process
- `get_changeRecords(...)`:
 - actual records exchange
 - performed by the node which has just received a notify (records pull)

UBR Coordination Mechanism

- Based on pure replication (every federated node propagates the own entries to the others)
- Circular one-way exchange of information performed in two rounds
- Process performed daily



- Static nature of data
- A few (rich) participants
→ limited inconsistency

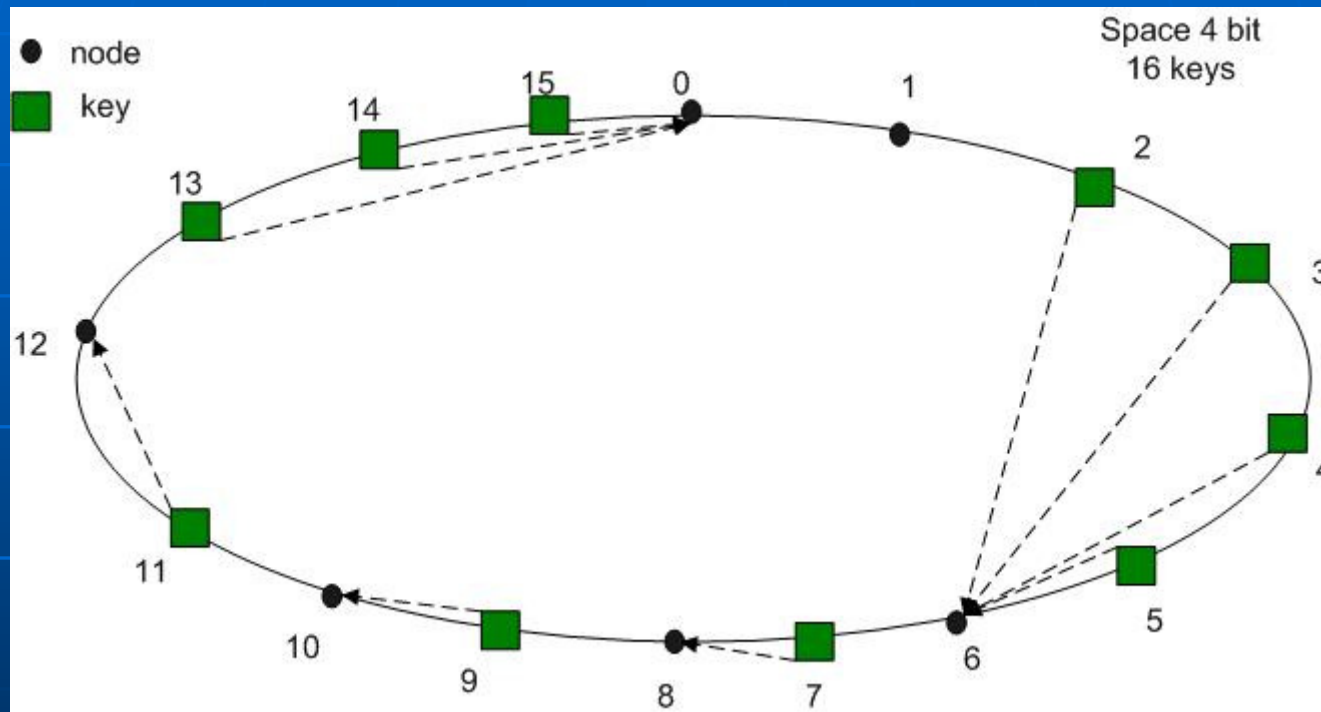
Another mechanism?

- The pure replication (in two rounds) implies a huge overhead in case of a large number of participants
- Small and Medium organizations are excluded
- A new lightweight and efficient mechanism may allow another concept of UDDI Cloud (with thousand of registries)
- Constraints:
 - Preservation of compatibility with UDDI standard
 - Coordination mechanism without a full entries propagation
 - Low performance impact

Distributed Hash Table (DHT) adoption

- Hash Table distributed among a set of cooperative nodes
- Insertion and lookup operations
- Autonomous, dynamic, and fault-tolerant
- Working model:
 - Large, uniform keys space through a hash function (e.g. SHA-1) mapping data with m bit (160 in case of SHA-1)
 - Keys and node IDs are mapped in the same space
 - Keys and node IDs are handled as points of a circular ring (from 0 to 2^m)
 - “A Key belongs to its successor, the first encountered node following the clockwise direction” (Key-based routing)
 - Topology preserved dynamically in case of joins, departures, fails through Stabilization algorithm
 - Configurable degree of data replication on neighbors
 - Maximum routing table size and maximum number of hops tends to $O(\log(n))$

DHT – Keys storage



DHT - Routing tables (1)

i = node; N = max num of node; m = num of bit; k = configuration parameter

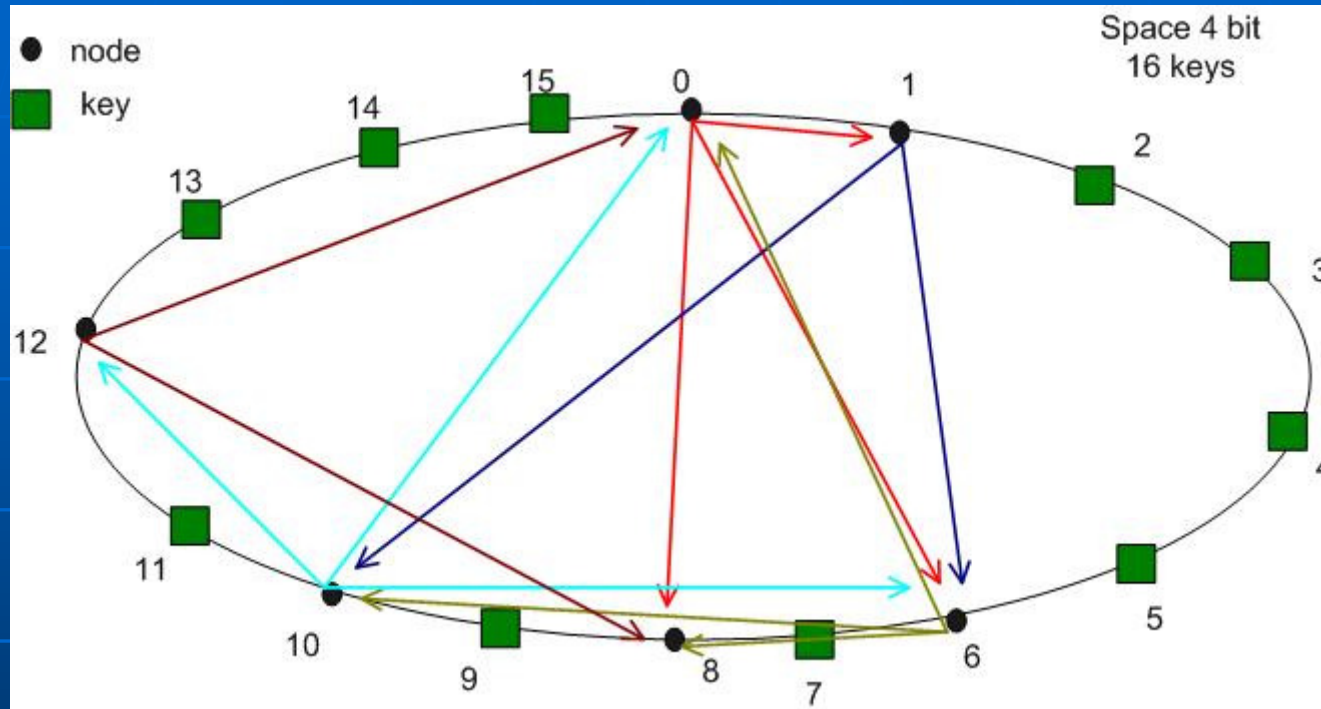
Routing table for node i : $(i + 2^b) \bmod(N)$

where: $0 \leq i \leq N-1$; $N=2^m$; $0 \leq b < \log_k(N)$ and i, N, m, b, k integers

With $k = 2$; $m = 4 \Rightarrow N = 16$, $b = 4$

	$i = \underline{0}$	$i = \underline{1}$	$i = \underline{6}$	$i = \underline{8}$	$i = \underline{10}$	$i = \underline{12}$
$i+2^0$	$0+1 = \underline{1}$	$1+1 = 2$ $\rightarrow \underline{6}$	$6+1 = 7$ $\rightarrow \underline{8}$	$8+1 = 9$ $\rightarrow \underline{10}$	$10+1 =$ $11 \rightarrow \underline{12}$	$12+1 = 13$ $\rightarrow \underline{0}$
$i+2^1$	$0+2 = 3$ $\rightarrow \underline{6}$	$1+2 = 3$ $\rightarrow \underline{6}$	$6+2 = 8$ $\rightarrow \underline{8}$	$8+2 = \underline{10}$	$10+2 =$ $\underline{12}$	$12+2 = 14$ $\rightarrow \underline{0}$
$i+2^2$	$0+4 = 4$ $\rightarrow \underline{6}$	$1+4 = 5$ $\rightarrow \underline{6}$	$6+4 = 4$ $\rightarrow \underline{10}$	$8+4 = \underline{12}$	$10+4 =$ $14 \rightarrow \underline{0}$	$12+4 = \underline{0}$
$i+2^3$	$0+8 = \underline{8}$	$1+8 = 9$ $\rightarrow \underline{10}$	$6+8 = 14$ $\rightarrow \underline{0}$	$8+8 = \underline{0}$	$10+8 = 2$ $\rightarrow \underline{6}$	$12+8 = \underline{8}$

DHT – Routing tables (2)



Maximum routing table size: $(k-1) \cdot \log_k(N)$

Maximum number of (virtual) hops: $\log_k(N)$

In general $K = 2$

Replication APIs mapped onto a DHT

- `notify_changeRecordsAvailable(...)`: stores in the DHT a pair formed by `<UDDI data structure (e.g. binding Template, businessEntity) - UDDI URL>`
- `get_changeRecords(...)`: creates a “redirect” entry in the database which is used to perform a search in the DHT when a client looking for a record does not find anything matching the requested one

Considerations

- Global number of entries :

- Pure replication mechanism = $n * p$

- DHT based replication = $k * p$

`n = # registries; p = average # entries per registry;`

`k = degree of replication (constant)`

- Global generated traffic:

- Pure replication mechanism $\sim O(n^2)$

- DHT based replication $\sim O(\log(n))$

Conclusions and Future Works

- The compatibility with the UDDI standard is preserved
- It is required a slight modification of already developed UDDI implementation:
 - Overload of Replication APIs
 - Plug-in of an ad-hoc module reacting when an entry is not found and performing the research in the DHT
- Every registry has to host a DHT portion
- A concrete performance evaluation
- Integration of WS protocols in the DHT transport layer