



# IoT: lecture 3

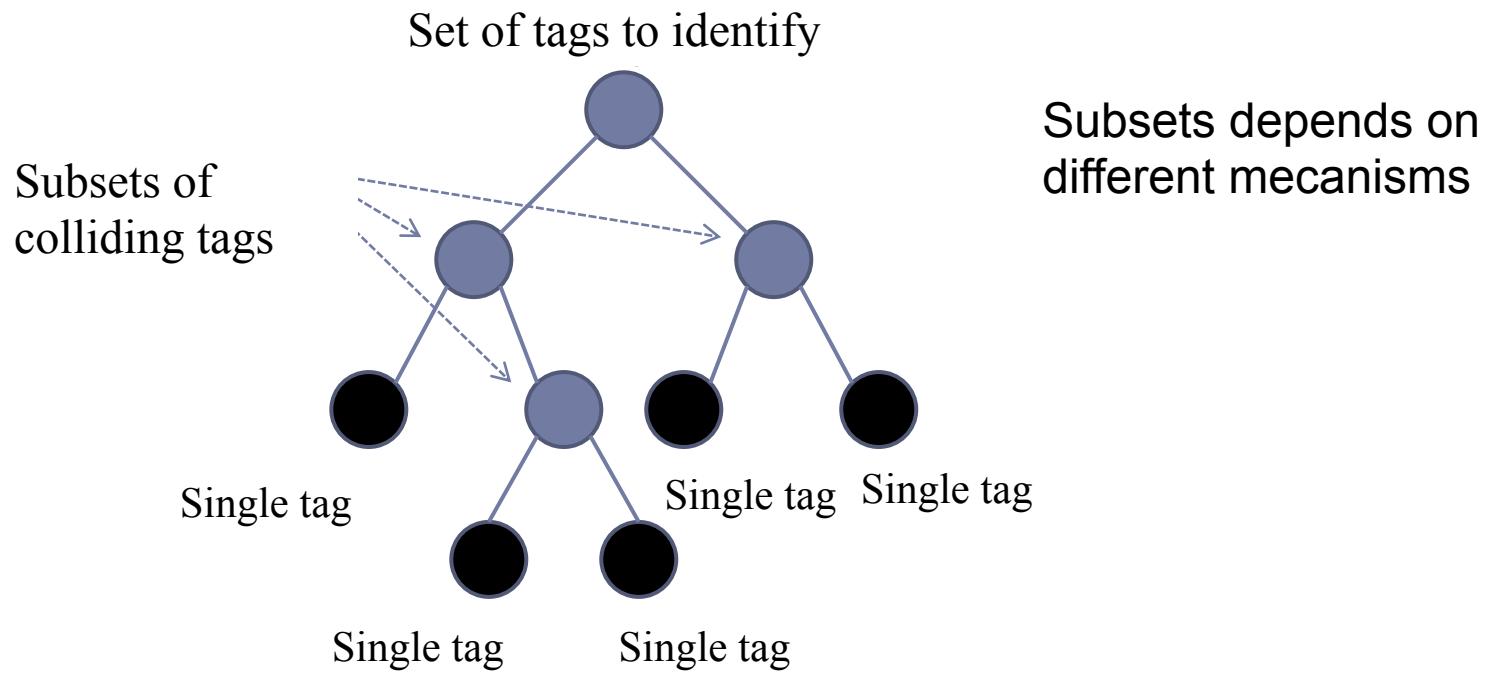
**Gaia Maselli**  
Dept. of Computer Science

# Tree-based protocols for tag identification

# Tree based protocols



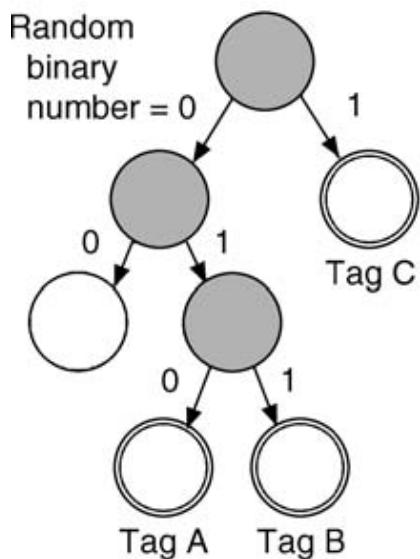
- To search unique tag ID (EPC) they follow a binary tree structure
  - Root node: Initial set of tags (to be identified)
  - Intermediate nodes: groups of colliding tags
  - Leaf nodes: identified tags



# Tree based: Binary splitting (BS)



- Tags are grouped based on the generation (inside tags) of a random binary number



- No tag transmission
- One tag transmission
- Tag collision

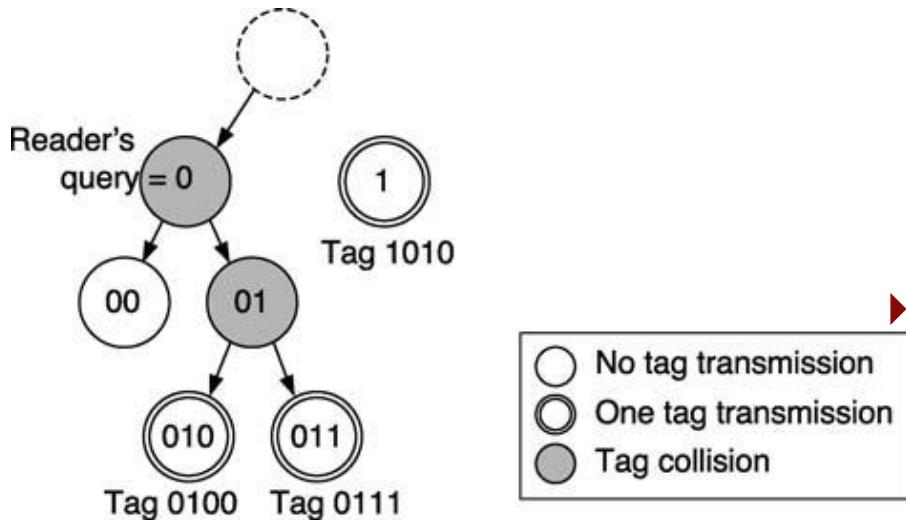
- Tags have a counter initialized to 0
- Tags transmit when their counter is 0
- The reader notified the tags about the query outcome (identification, collision, no answer)
- Tags update their counter based on the query outcome
  - Collision: silent tags increase their counter by 1 while transmitting tags generate a random binary number (0,1) and sum it to the counter
  - No collision (identification or empty): every tags decrease the counter by 1

# Tree based: Query Tree (QT)



- Tags are grouped (and queried) according to the generation of a random binary number

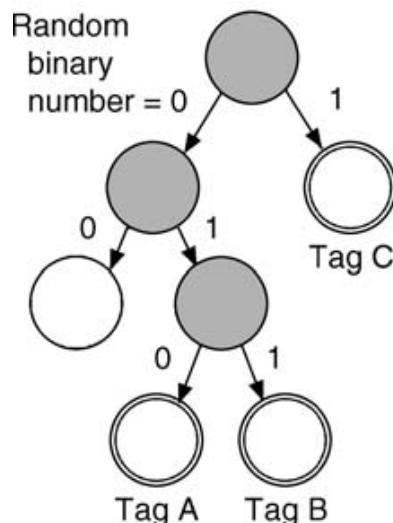
- The reader sends a query containing a binary string
- The tags that match the string with a prefix ID reply with their ID
- If there is a collision on the string  $q_1 q_2 \dots q_x$  ( $q_i \in \{0, 1\}$ ),  $1 \leq x < b$ , and  $b$  is the number of bits in the ID, the reader append one bit (0 and 1) at the string and send two new queries  $q_1 q_2 \dots q_x 0$  and  $q_1 q_2 \dots q_x 1$
- Colliding tags are then split into two subset



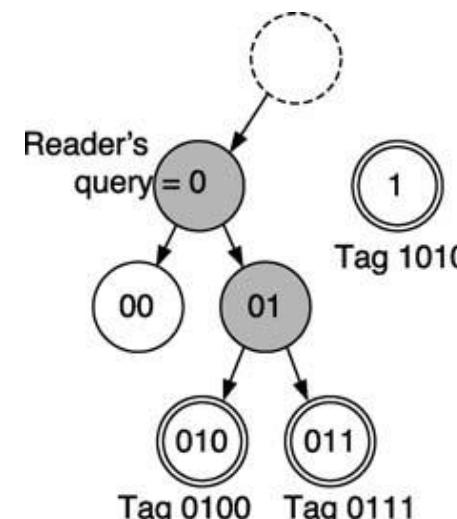
# Binary Splitting vs Query Tree



- Identification trees are similar
- The generation of random numbers inside a BS tags reflect the generation of random bits on QT queries



Binary Splitting (BS)



Query Tree (QT)

# Query Tree Improved (QTI)



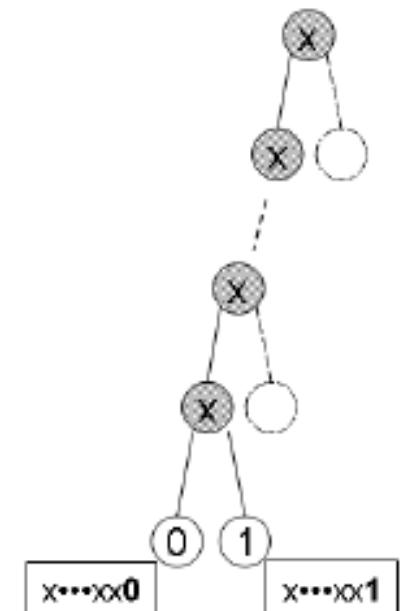
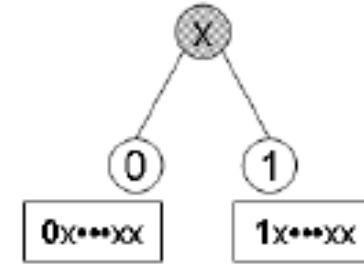
SAPIENZA  
UNIVERSITÀ DI ROMA

- Optimization of the number of queries, avoiding obvious queries
- Example:
  - Query with prefix “p” causes collision
  - Query with prefix “p0” causes no answer
  - Query with prefix “p1” is skipped because it will cause a collision, and “p10” and “p11” are queried next

# Effect of ID distribution on QT

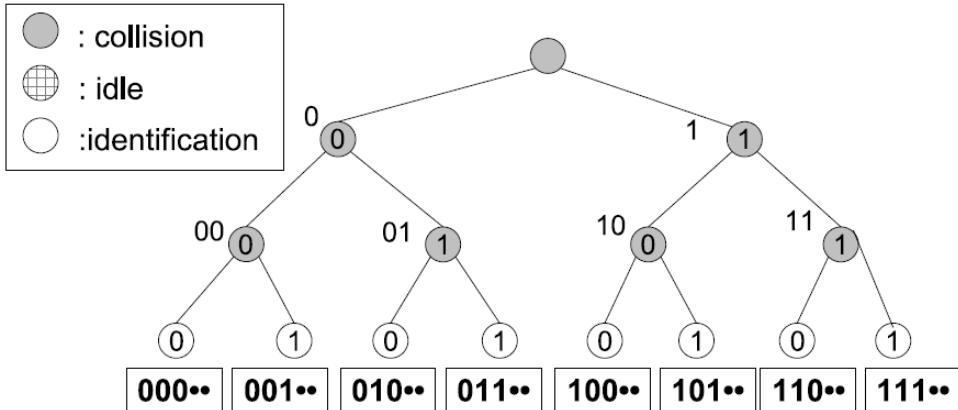


- Best ID distribution: the idea is to minimize the number of collisions (shortest common prefix).
  - In the case of two tags: if their IDs differ for the most significant bit
    - < 00000 >
    - < 10000 >
  - the inventory will result in only one collision (which is the minimum number of collisions to identify two tags).
  
- Worst ID distribution: the idea is to maximize the number of collisions (longest common prefix)
  - In the case of two tags: if they differ for the least significant bit:
    - < 00000 >
    - < 00001 >
  - The inventory will result in as many collisions as the common bits in the IDs



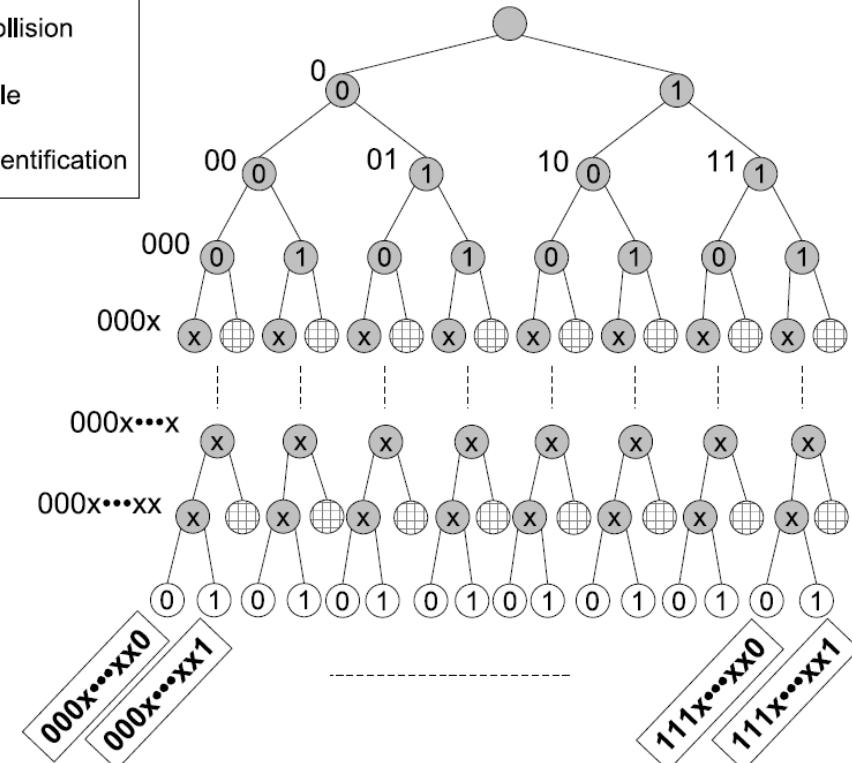
# Effect of ID distribution on QT (cont)

## ■ Optimal distribution



example

- : collision
- ▨ : idle
- : identification



example

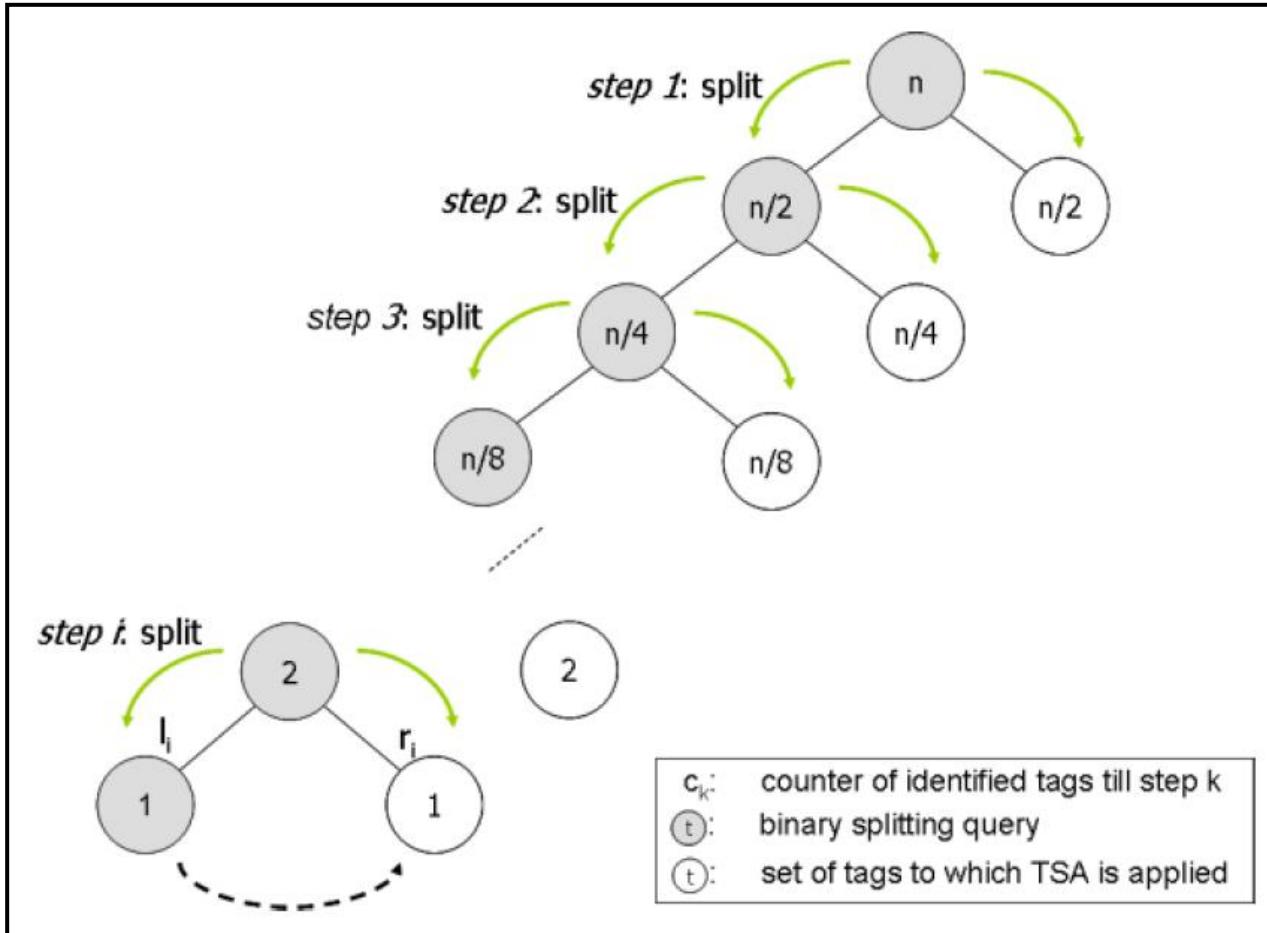


# A hybrid protocol: Binary Splitting Tree Slotted Aloha

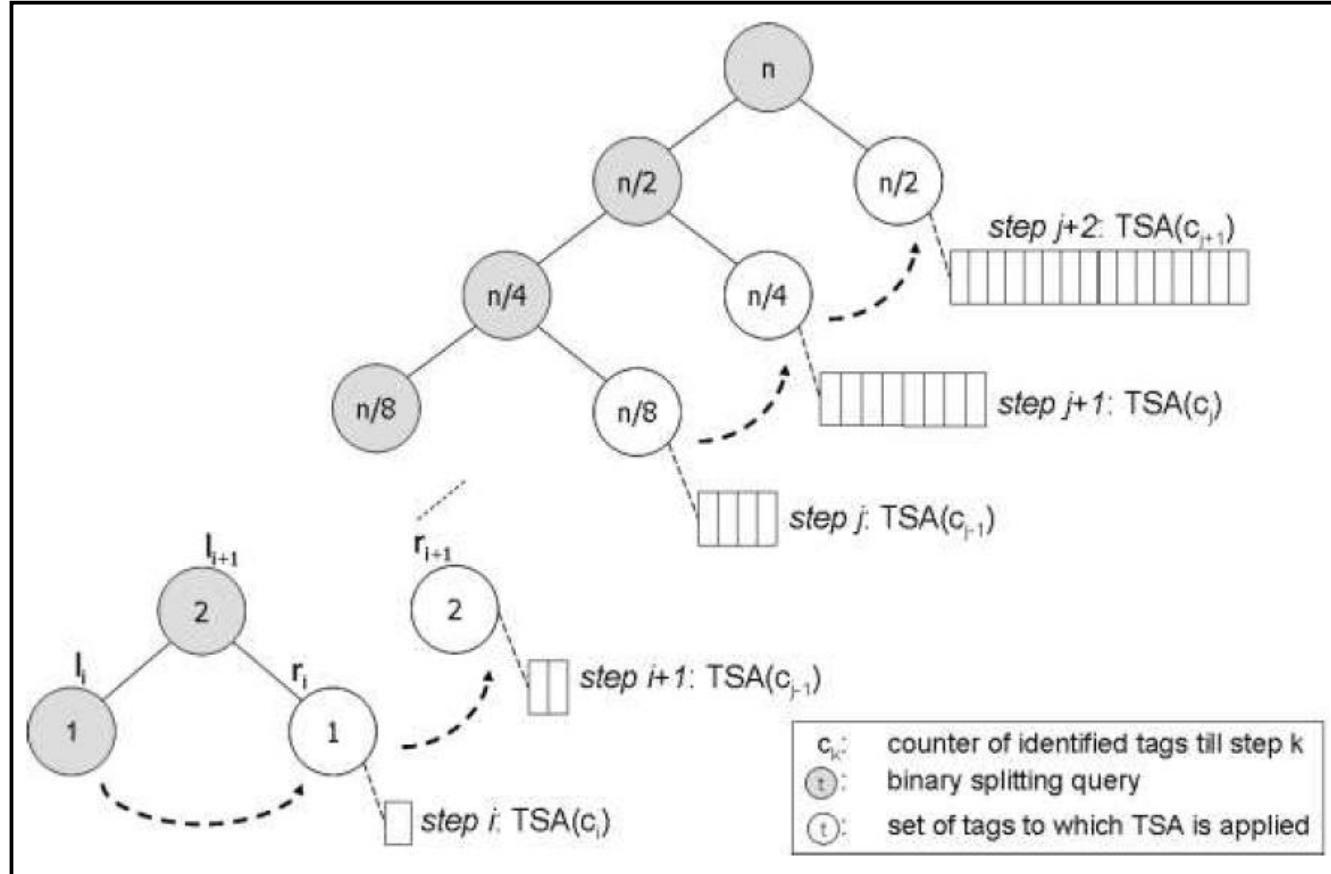
# Binary Splitting to estimate tags



SAPIENZA  
UNIVERSITÀ DI ROMA

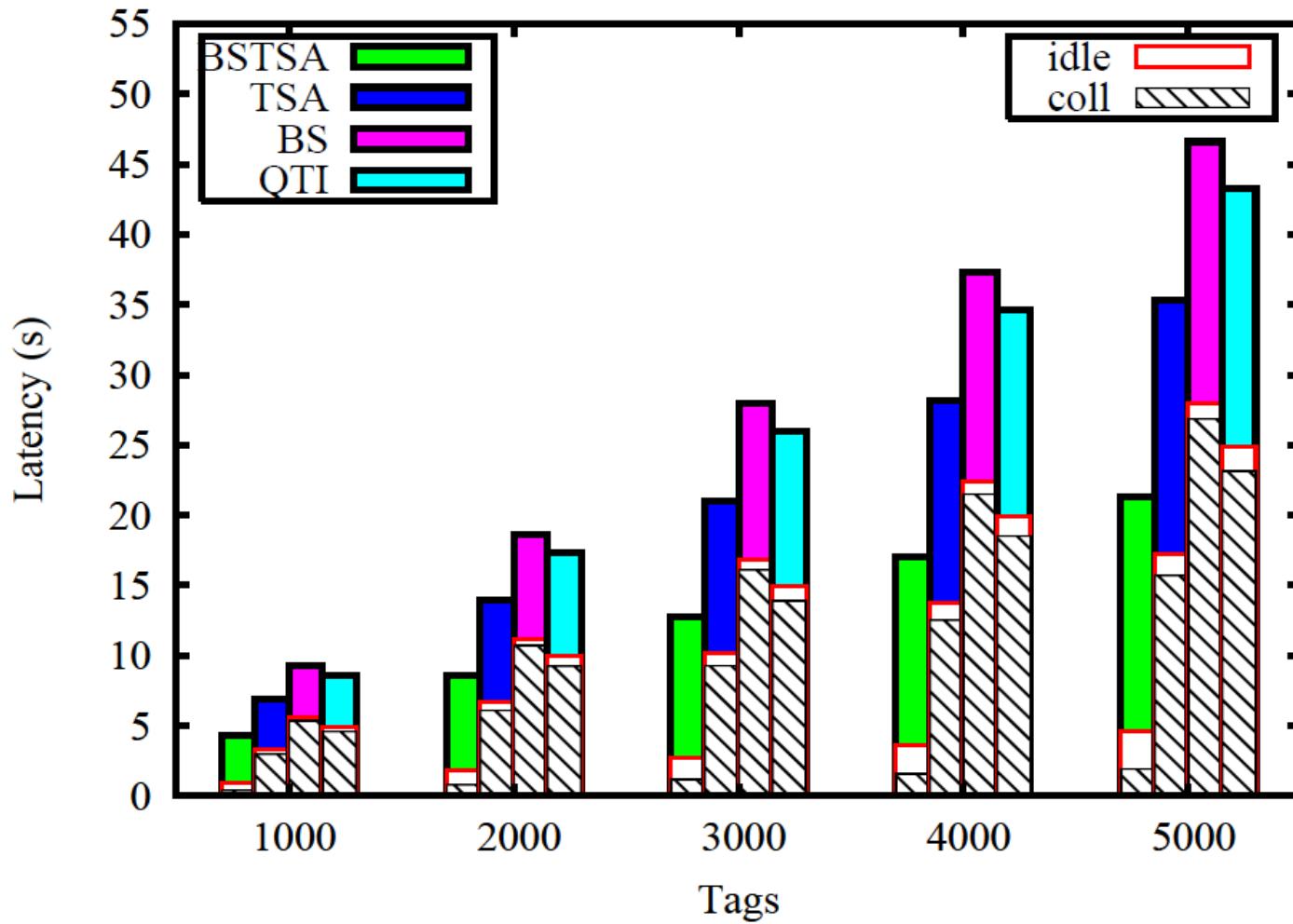


# Tree Slotted Aloha to identify tags



## Tree Slotted Aloha phase

# BSTSA performance





# Readings

- Papers available on IEEE and ACM digital libraries:
- T.F. La Porta, G. Maselli, C. Petrioli, "**Anti-collision Protocols for Single-Reader RFID Systems: Temporal Analysis and Optimization**", *IEEE Transactions on Mobile Computing*, vol.10, no. 2, pp.267,279, Feb. 2011.