MobiHide:
A Mobilea Peer-to-Peer System
for
Anonymous Location-Based Queries

Written By:
Gabriel Ghinita, Panos Kalnis and Spiros Skiadopoulos

Presented By:
Laurynas Šikšnys

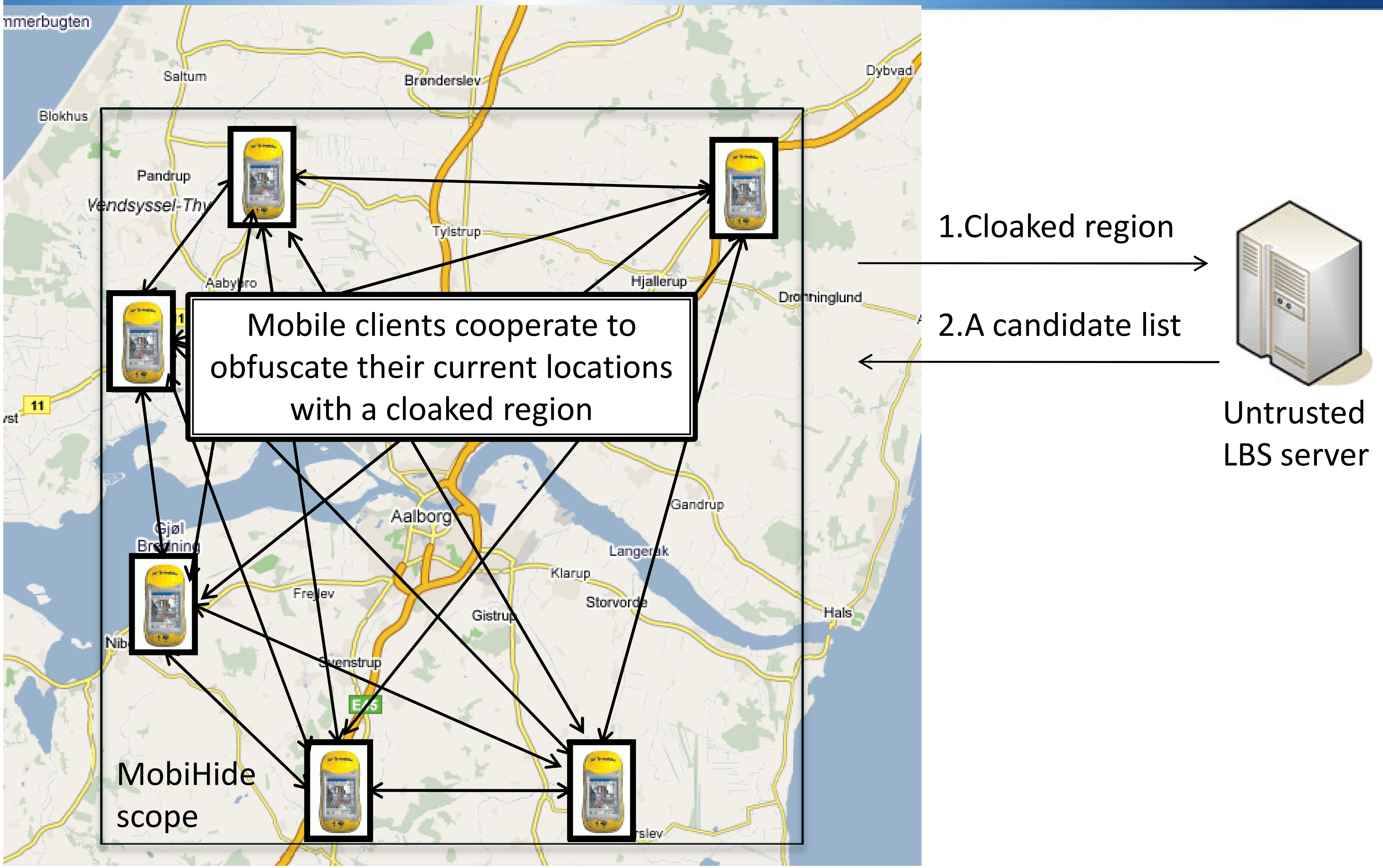
MobiHide: The Overview

- Overview
- Concept
- Architecture
- Implementation
- Performance
- Related Work
- Evaluation & Conclusions

MobiHide:The Concept

- Overview
- Concept
- Architecture
- Implementation
- Performance
- Related Work
- Evaluation & Conclusions

MobiHide:The Concept



MobiHide:The Concept

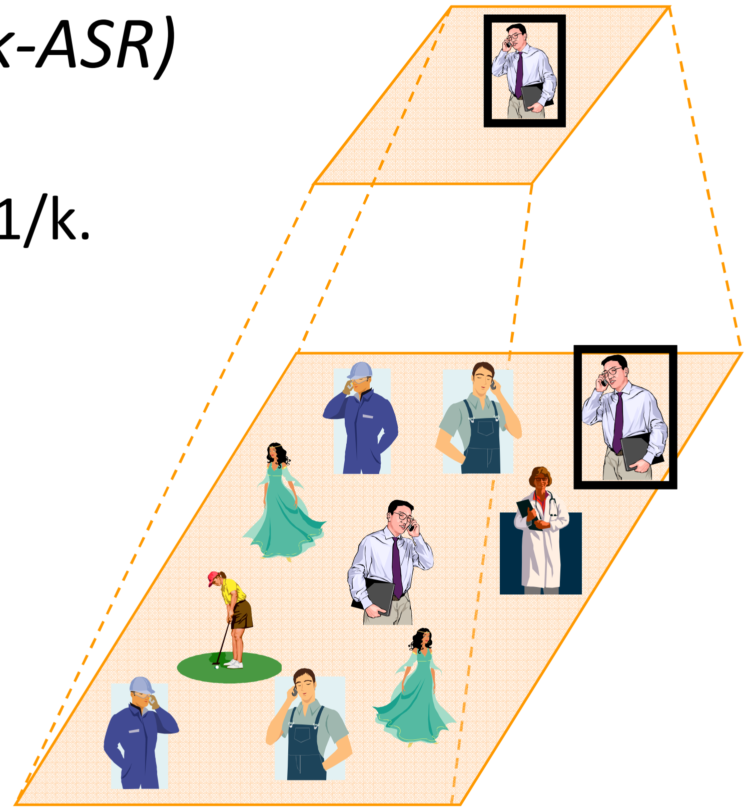
- The MobiHide – a peer-to-peer system, that provides cloaked region construction, and offers:
  - Strong anonymity – achieved by K-anonimity and Hilbert ordering
  - Fault tolerance;
  - Scalability;
  - High Performance;

  – achieved by proposed approach, based on Chord

MobiHide:The Concept

K-anonimity

- k-Anonymizing Spatial Region *(k-ASR)*
  - Encloses at least k users;
  - User identification probability ≤ 1/k.

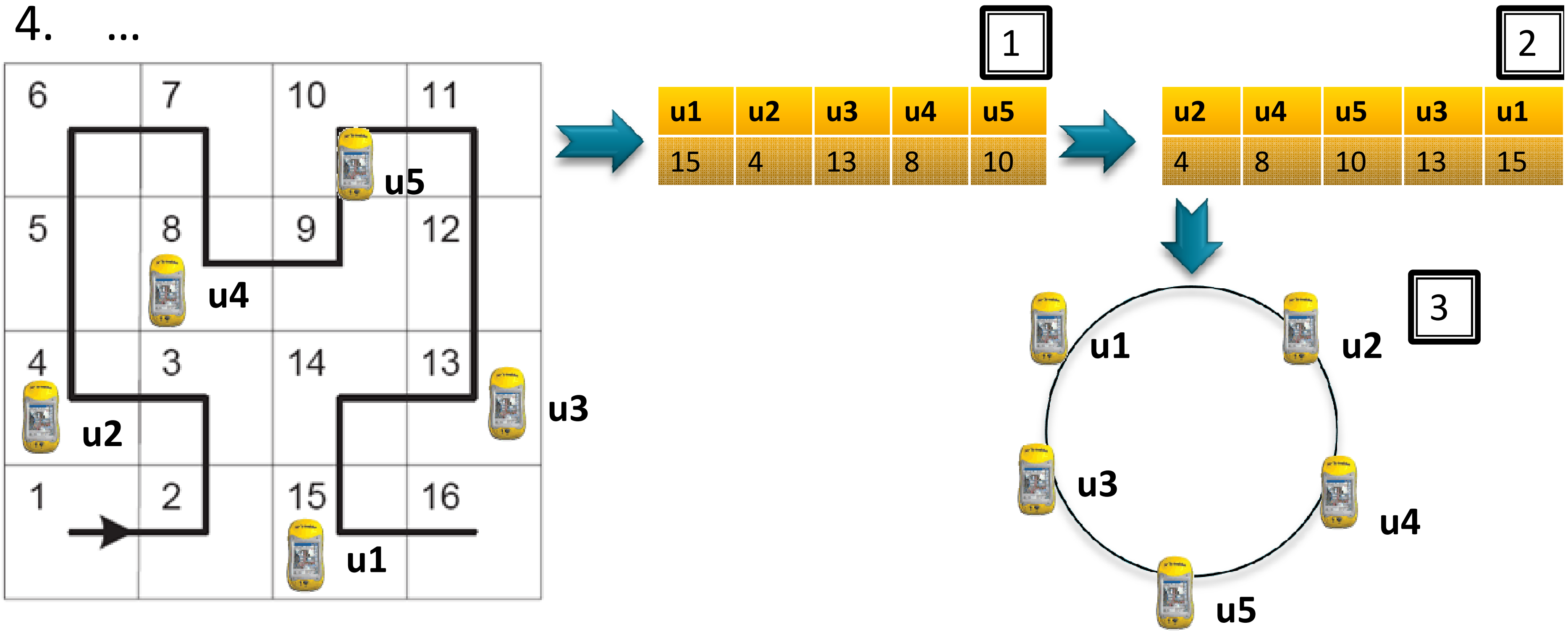


10-anonymity

MobiHide:The Concept

Anonymization Algorithm

1. Each user is assigned H(u)
2. Users are sorted according their H(u)
3. The sorted list is cycled
4. …



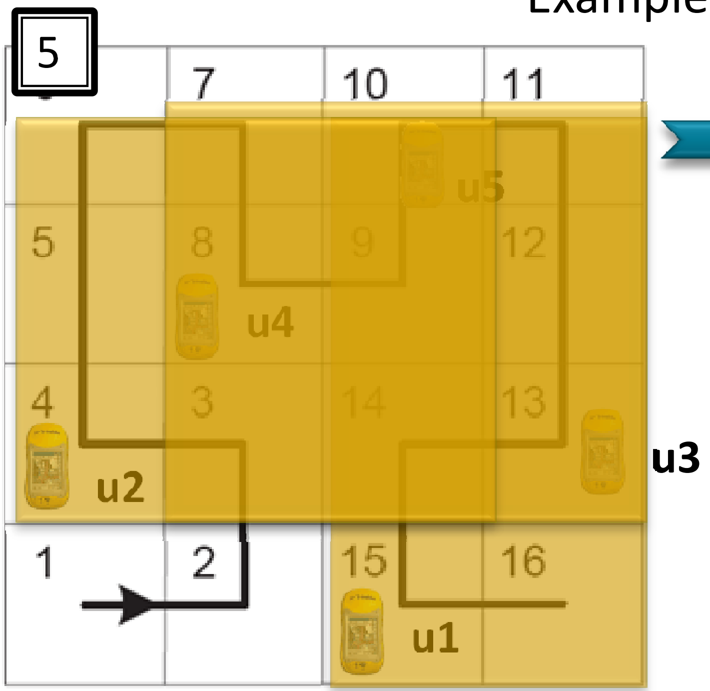
MobiHide:The Concept

Anonymization Algorithm (continued)

…

4. Consecutive K users are selected
5. K-ASR is constructed by finding MBR

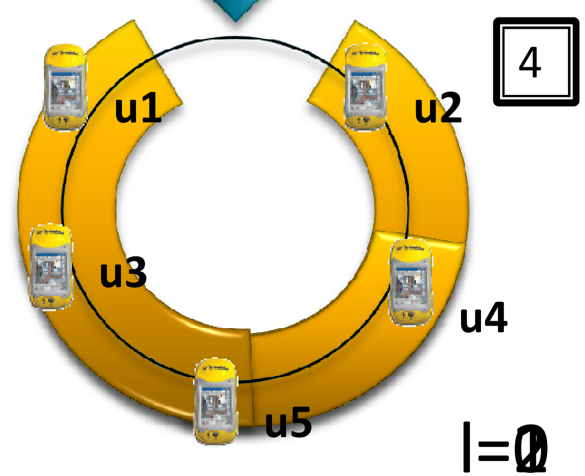
Example 3-ASR construction, issued by u5



u1	u2	u3	u4	u5
15	4	13	8	10

u2	u4	u5	u3	u1
4	8	10	13	15

K=3



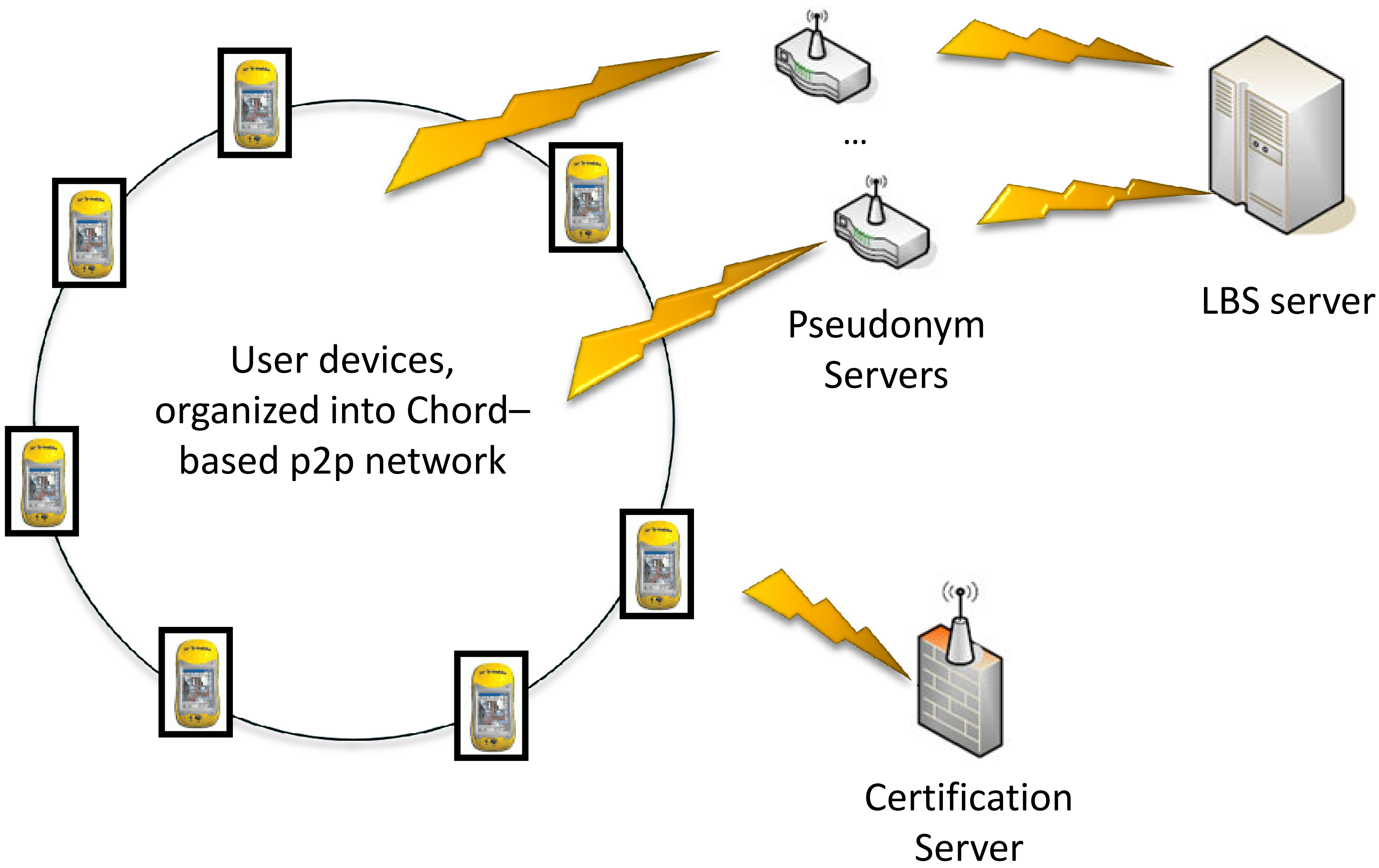
MobiHide:The Concept

- Anonymization Algorithm :
  - Guarantees anonymity for uniform query distribution;
  - Provided experimental probability of identifying querying user close to 1/K, when query distribution is skewed;

MobiHide: The Architecture

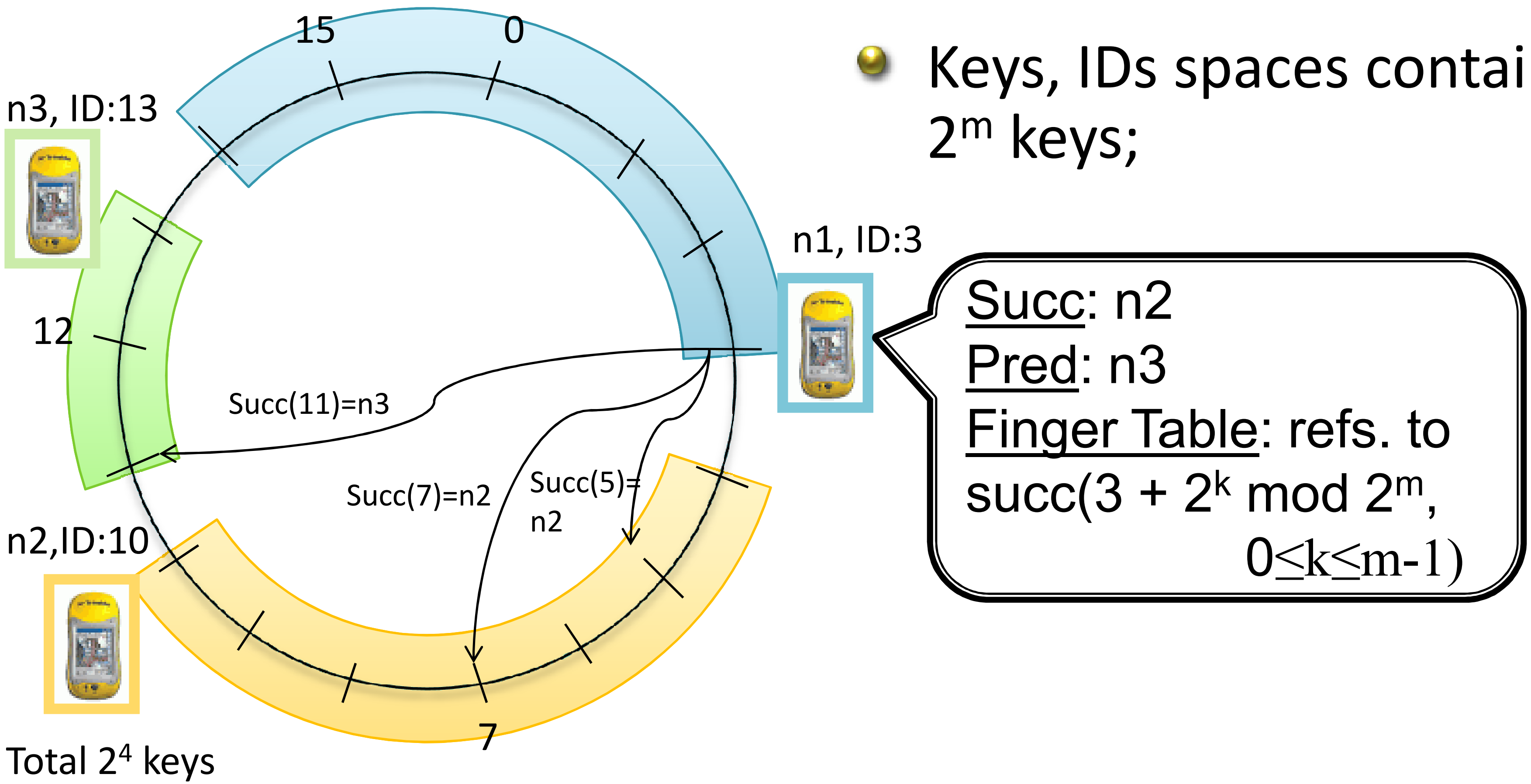
- Overview
- Concept
- Architecture
- Implementation
- Performance
- Related Work
- Evaluation & Conclusions

MobiHide: The Architecture



MobiHide: The Architecture

- Chord supports only 1 operation:
  "Given a key, map it onto node".
- This operation can also be used to find a node with id on the network.



- Keys, IDs spaces contains 2^m keys;

MobiHide: The Architecture

- Chord provides:
  - Simplicity
  - Good performance
  - Correct behavior
  - Scalability

MobiHide: The Implementation

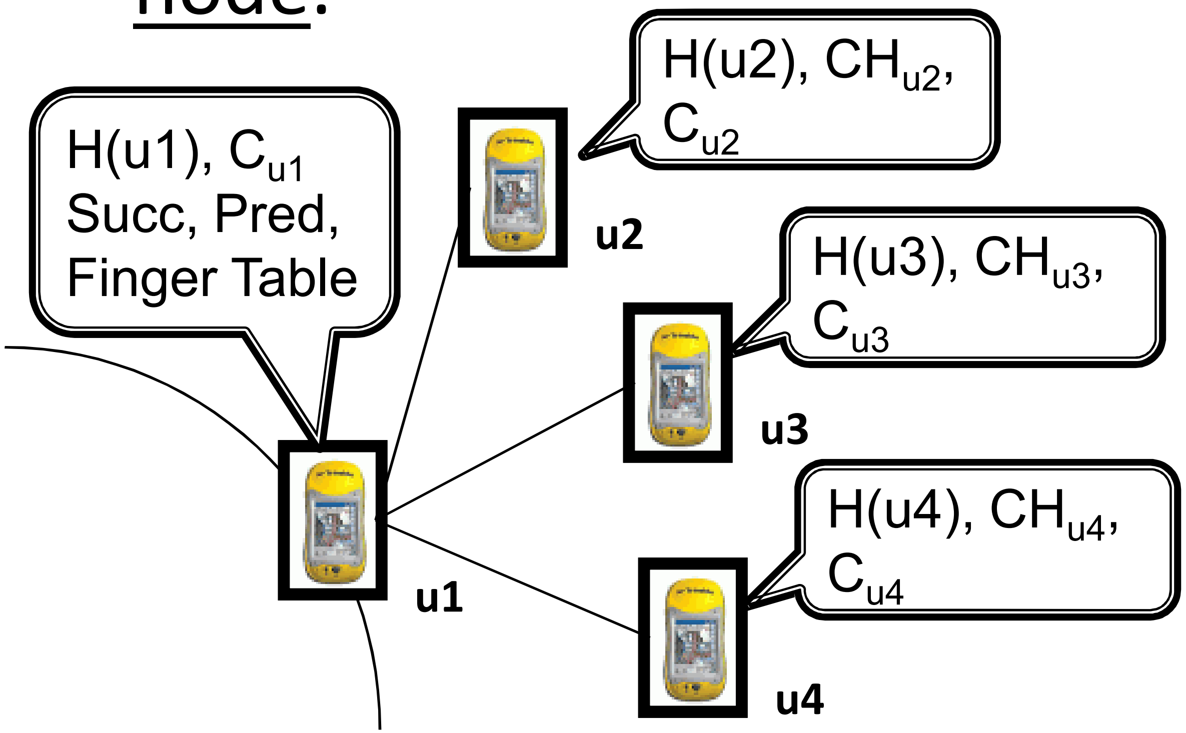
- Overview
- Concept
- Architecture
- Implementation
- Performance
- Related Work
- Evaluation & Conclusions

MobiHide: The Implementation

- Implemented operation performance is measured in term of:
  - - **latency**: the number of overlay hops on the longest path followed;
  - - **cost**: the number of transmitted messages.

MobiHide: The Implementation

- Idea Nr. 1: Map each user to distinct Chord node.
  - K-ASK construction latency is O(K) overlay hops.
- Idea Nr. 2: Assign a cluster of users to single Chord node.



- H(u) – Hilbert value of user u;
- C_u – cluster, that contains user u;
- CH_u – a head of cluster C_u.

Each cluster has from **α** to **3α-1** users, where **α** – a system parameter;

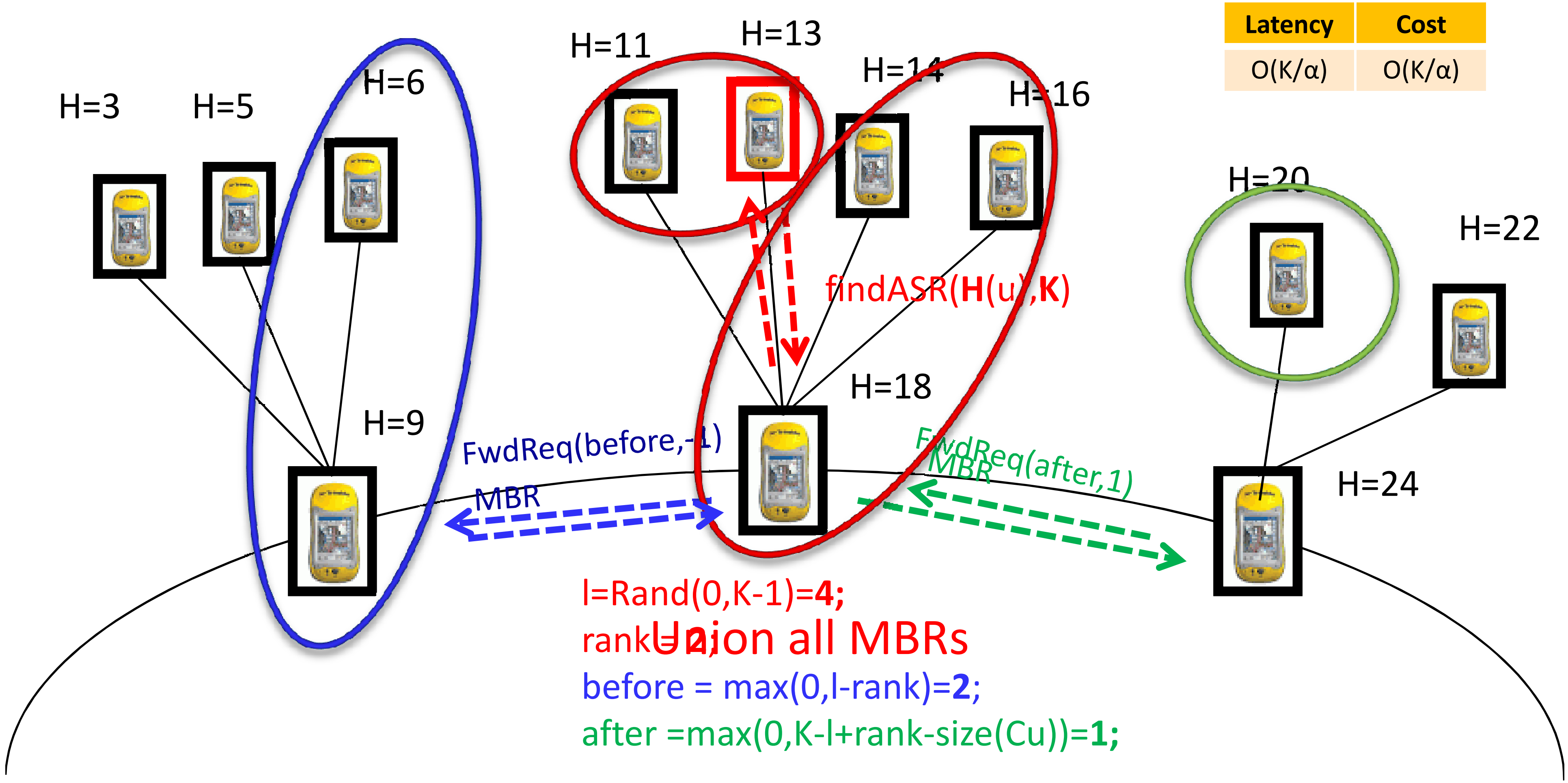
Heads are rotated periodically, when certain load threshold is reached.

MobiHide: The Implementation

- K-ASR generation algorithm

Example 8-ASR construction, issued by user with H(u)=13, α=2

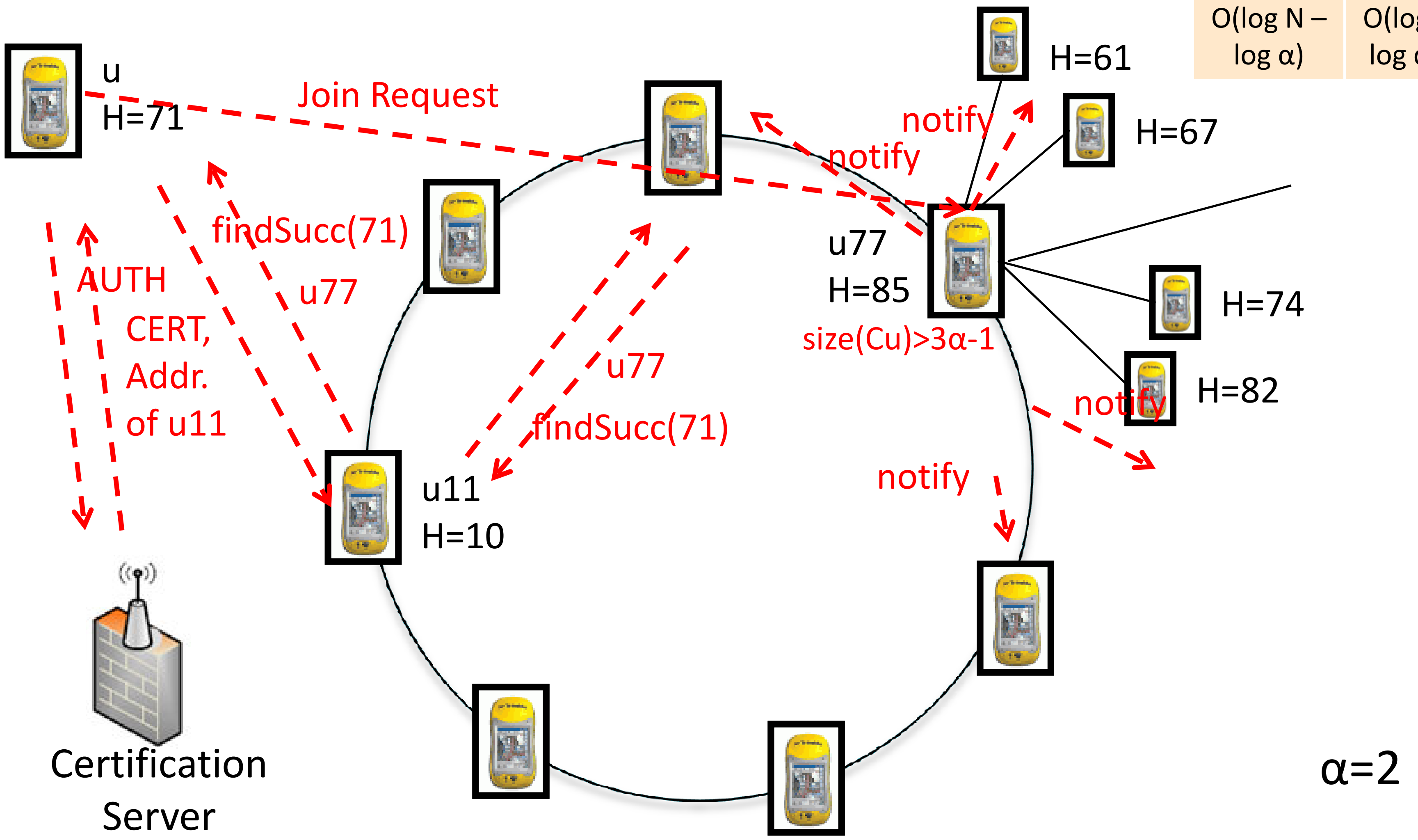
Latency	Cost
O(K/α)	O(K/α)



MobiHide: The Implementation

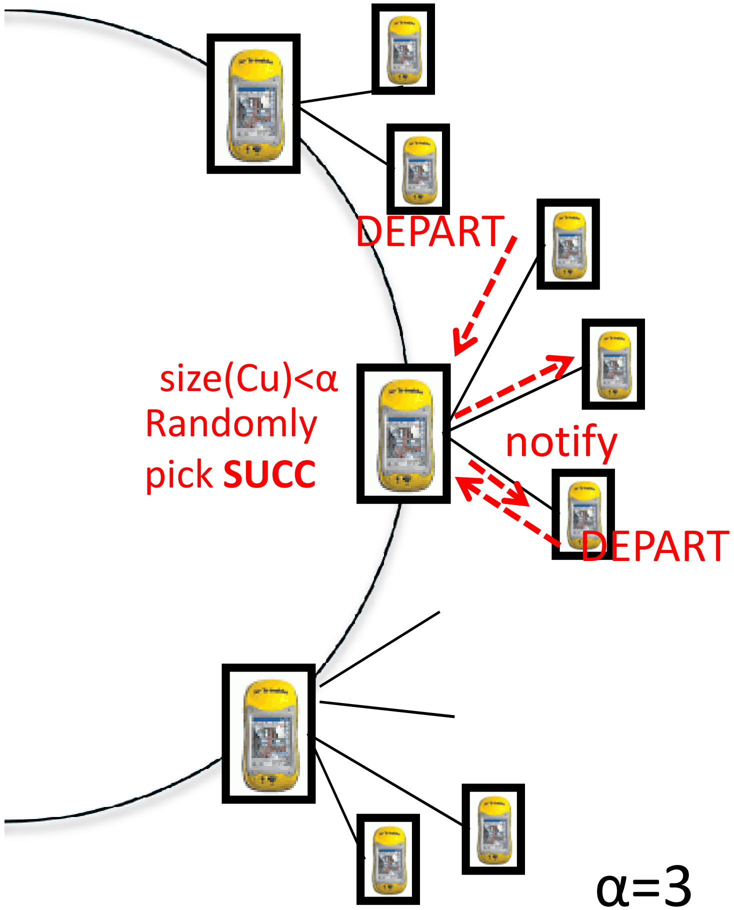
- Node Join Algorithm

Latency	Cost
O(log N – log α)	O(log N – log α+α)



MobiHide: The Implementation

- Node Depart Algorithm
  - Gracefull depart
  - Failure depart



- Intra-cluster maintenance
  - Beacon messages at σt
  - Failure is detected after 2σt elapsed with no response

MobiHide: The Implementation

- Realocation
  - If H(u) falls within key range of C_u, only CH_u is informed;
  - Otherwise, a graceful departure & join is performed;

Latency	Cost
$O(\log N - \log \alpha)$	$O(\log N - \log \alpha + \alpha)$

MobiHide: The Performance

- Overview
- Concept
- Architecture
- Implementation
- Performance
- Related Work
- Evaluation & Conclusions

MobiHide: The Performance

- MobiHide is compared with existing systems:
  - CloakP2P;
  - Prive.

MobiHide: The Performance

- Experimental performance evaluation
  - The setting of experiments
    - p2psim – packet level simulator for P2P systems [1]
      - Topologies with RTT = 1sec.
      - No link failures.
      - Limited lengths of packet queues on nodes.
    - Network-based Generator of Moving Objects [3]
      - Dataset corresponds to the San Francisco Bay Area
    - Parameter values:
      - <u>Client Count:</u> N = 1k .. 10k;
      - <u>Anonymity Level:</u> K = 10 .. 160;
      - <u>Cluster Size:</u> α = 5.

MobiHide: The Performance

- Anonymization Strength Experiment ("Center-of-K-ASR" attack)
  - Settings:N=10k; count(K-ASR gen)=10k;

  Query distribution ZipFian, v=0.8.

  - Results:

u_c – closest user to K-ASR center;
u_q – user, issued the query.
P(IdentifySource) =P($u_c = u_q$)

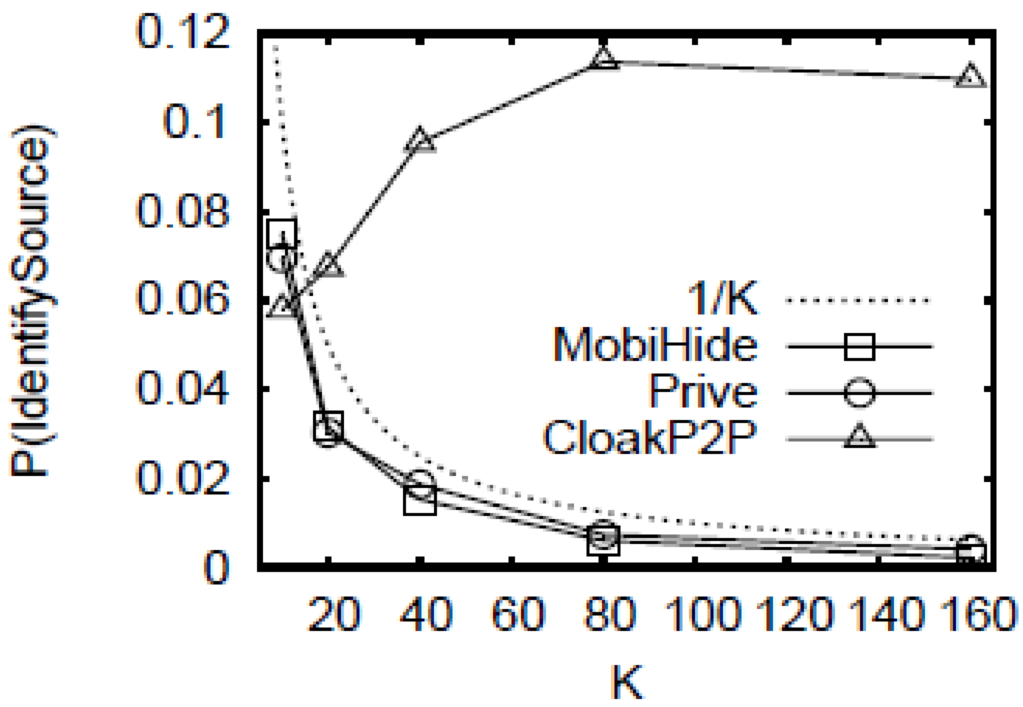


Fig. 10. "*center-of-K-ASR*" attack

Always P <=1/K

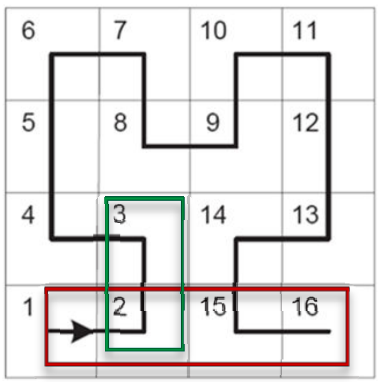
MobiHide: The Performance

- K-ASR Size Experiment

  (The Hilbert sequence wrapping impact on K-ASRs sizes)

  - Settings: N = 10k; query distributions: Uniform, ZipFian (v=0.8).
  - Results:

ASR Area – A percentage of the entire data space, covered by K-ASR.



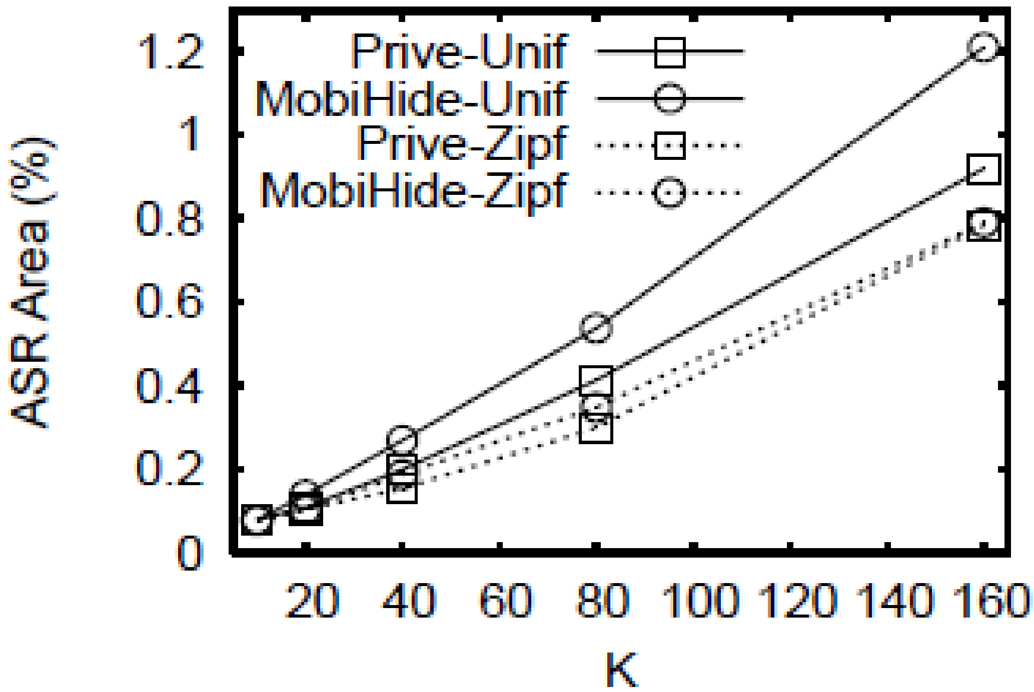


Fig. 12. K-ASR Area

Looses against Prive

MobiHide: The Performance

- Scalability (response time) Experiment
  - Settings: N = 1k, 5k, 10k; querying user is selected with distribution: ZipFian (v=0.8); processing time on node is exponentially distributed with mean 10ms.
  - Results:

QueryRate – Queries per User per Hour

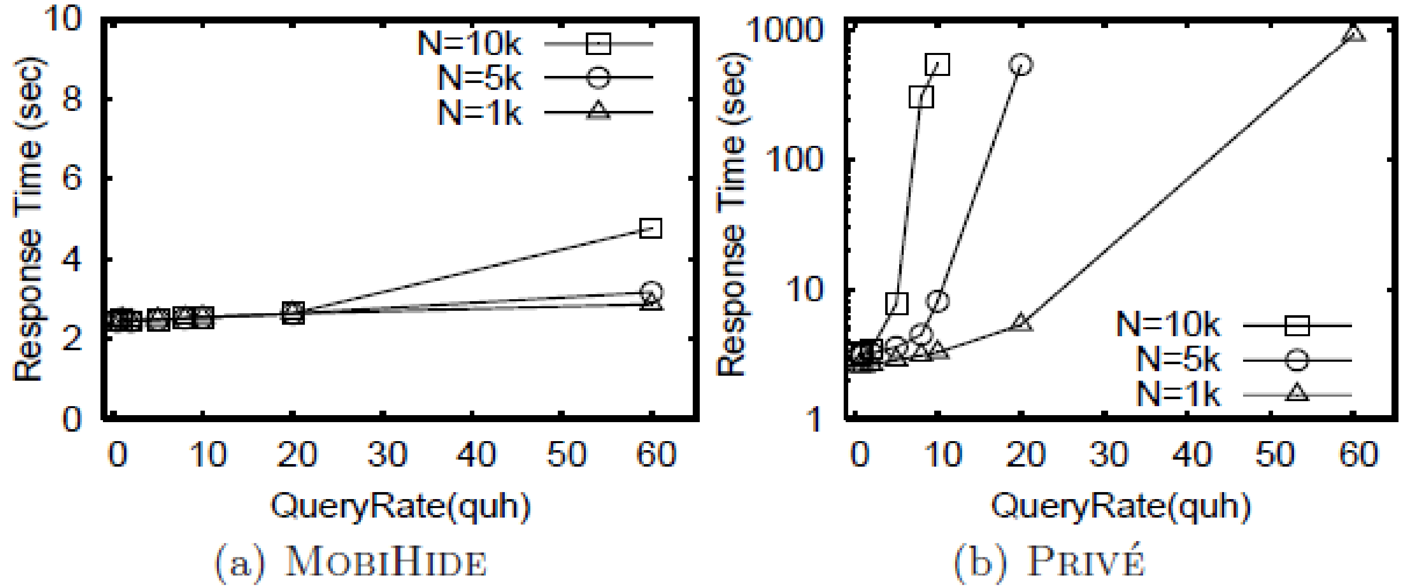


Fig. 13. Scalability, $\mathcal{K} = 40$

Beats Prive

MobiHide: The Performance

- Results of experiments with different α values
  - Node join experiment:

    High values of α yield low latency, however high communication costs.
  - K-ASR generation

    The system favors high values of α
- Conclusions
  - When setting α, a compromise among K-ASR generation, maintenance cost, scalability must be reached.
  - Suggestion for α : $5 < \alpha < 10$;

MobiHide: The Performance

- Load Balancing Experiment
  - Settings: N=10k; α=5; K=20; quh=3.6. t_{sym}=3hours; rt_{count}=300 messages.
  - Results:

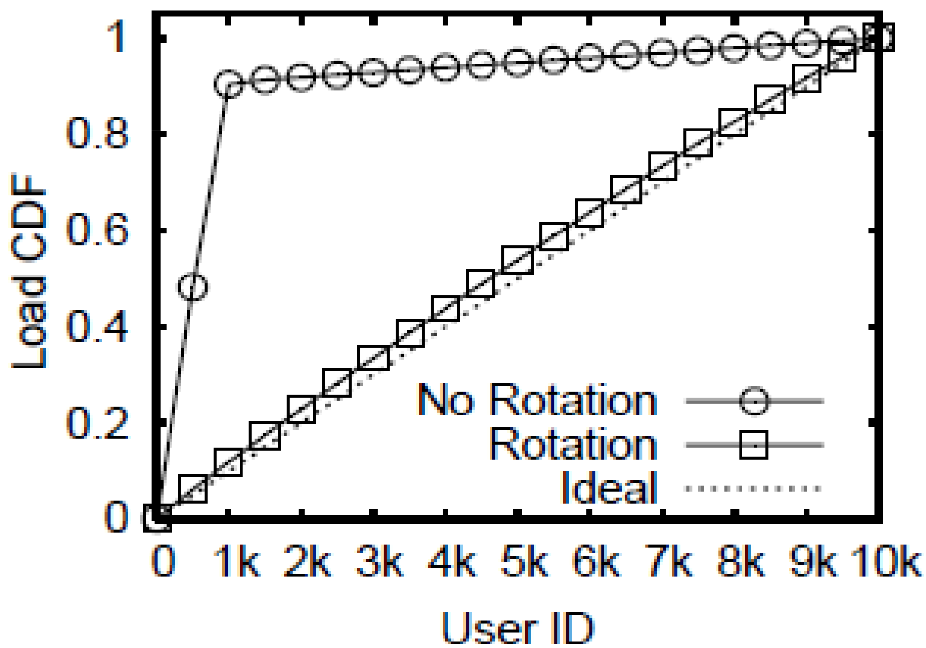


Fig. 16. Load Balancing

With rotation, the load balancing is very close to the ideal

MobiHide: The Performance

- Fault Tolerance Experiment
  - Settings: N=10k; α=5; σt (intra-cluster beacon time) = 10sec; t_{suc_pred} = 3sec; t_{suc_list} = 10sec; t_{ft} = 30sec; 25% users fail.
  - Results:

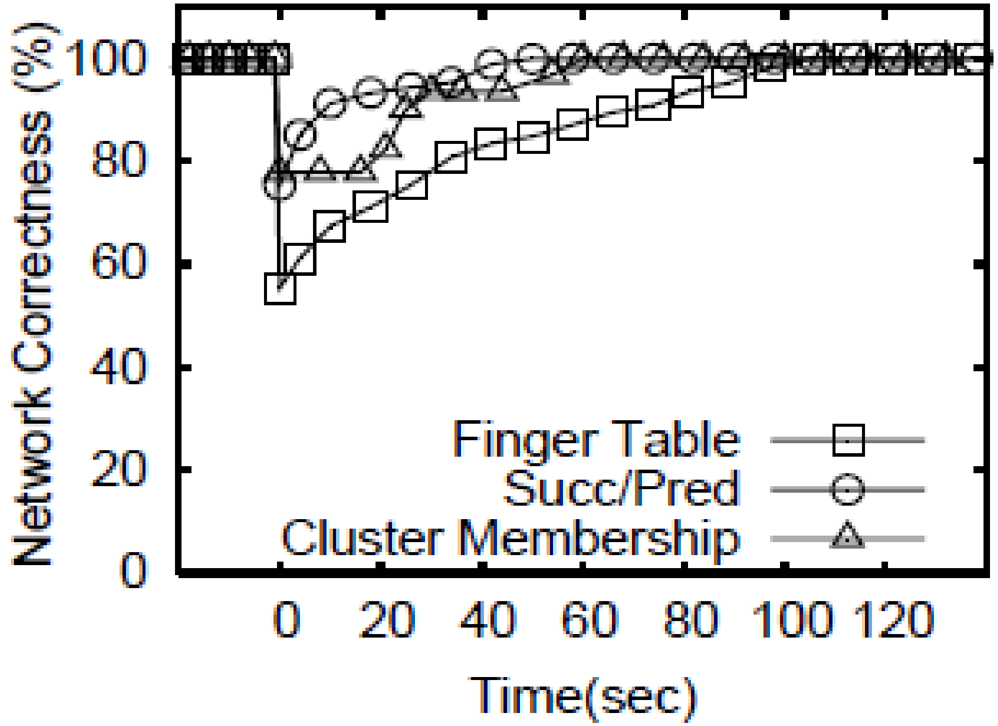


Fig. 17. Fault Tolerance

MobiHide: The Related Work

- Overview
- Concept
- Architecture
- Implementation
- Performance
- Related Work
- Evaluation & Conclusions

MobiHide: The Related Work

- The New Casper [15]
  - Limiting client-server architecture
- CloackP2P [5]
  - Fails to provide privacy for many user distributions
- Prive [8]
  - Suffers from slow response times
- Our semester project

MobiHide: Evaluation & Conclusions

- Overview
- Concept
- Architecture
- Implementation
- Performance
- Related Work
- Evaluation & Conclusions

MobiHide: Evaluation & Conclusions

- The approach
  - \+ Scalable
  - \+ Fully decentralized
  - \+ Fault-tolerant
  - \- No privacy guarantees for skewed query distributions
- The Paper
  - \+ Comprehensive performance study
  - \- Small error in K-ASR generation algorithm

MobiHide: Evaluation & Conclusions

```
u.findASR(H,K)
  compute rank_H in sorted order of C_u
  generate random offset l
  before = max(0,l - rank_H)
  after = max(0,K-l + rank_H − size(C_u))
  if (after > 0)
    succ.FwdReq(after,1)
  if (before > 0)
    pred.FwdReq(before,−1)
  wait for partial MBRs
  K-ASR = union of all received MBR

u.K-request(K)
  call CH_u.findASR(H(u),K)
u.FwdReq(count,direction)
  if (direction == 1) /*Look Forward*/
    return MBR of first count keys
    if (count > size(C_u))
      succ.FwdReq(count − size(C_u),1)
  else                /*Look Backward*/
    return MBR of last count keys
    if (count > size(C_u))
      pred.FwdReq(count − size(C_u),−1)
```

Fig. 7. Pseudocode for $\mathcal{K}$-Request

Thank you for your attention!