# Peer-to-Peer Collaborative Caching for Privacy-Preserving Location-based Services

Kangsoo Jung, Seog Park
Sogang University
{azure84, spark}@sogang.ac.kr

## 1. Problem definition

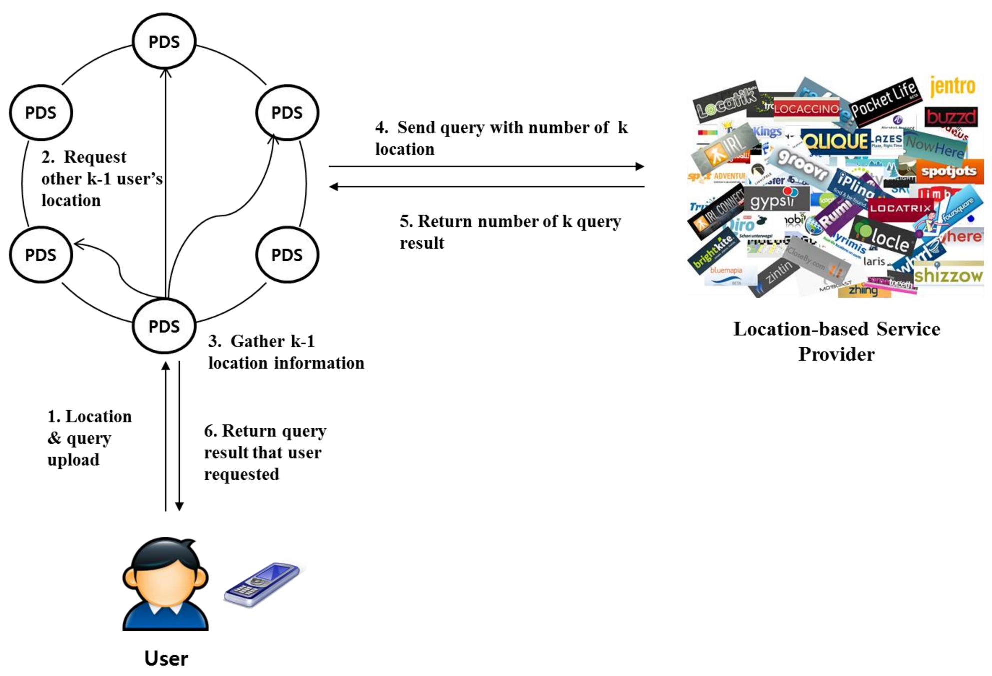
### Location privacy violation

- Collected location information can bring about privacy violations such as crime and unwanted advertisement.
- To prevent privacy violation, many research exists on preventing privacy violation(location k-anonymity, pseudonym).
- However, existing research is difficult to apply real-world LBS (Location-based Service) because it's data utility-privacy tradeoff.

### Our solution

- We propose collaborative caching technique which share query results among users to overcome the privacy-data utility tradeoff.
- Proposed technique is analyzed in terms of privacy, broadcasting cost and update cost.

## 2. Proposed architecture




**Step 1:** The user uploads his/her location information to the PDS (Personal Data Storage).
**Step 2:** The PDS requests the k-1 other PDS's addresses to meet the location k-anonymity.
**Step 3:** Other k-1 PDS returns their requested addresses to the PDS
**Step 4:** The PDS sends queries to the LBS provider with *k* location information.
**Step 5:** The LBS provider processes the query and returns the *k* query results to the PDS.
**Step 6:** The PDS filters the unnecessary query results and sends the correct result to the user.

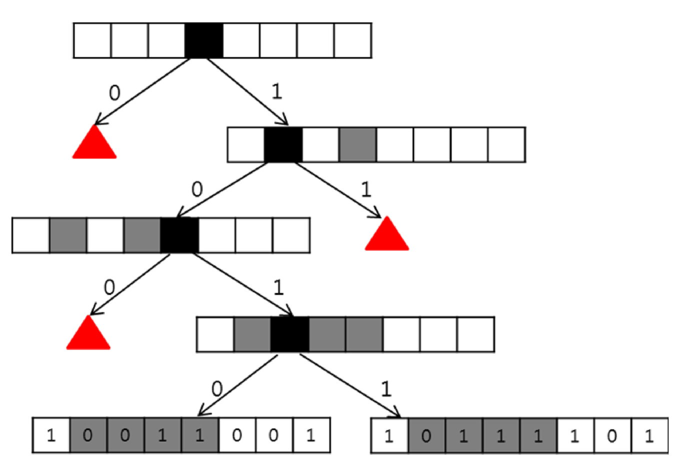
## 3. Collaborative caching technique

### Necessary of collaborative caching

- Location k-anonymity is used to reduce the possibility to infer a user's exact location.
- However, it generates an extra number of query results.
- To reduce an extra number of query results, we propose P2P collaborative caching technique based on counting bloom filter.

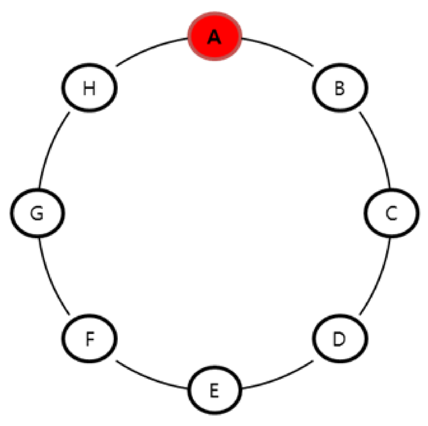
### PDS grouping algorithm

- Based on P2P, no trusted third party to manage entire query information. Thus, broadcasting cots is huge burdensome in P2P.
- We propose a PDS grouping algorithm for reducing broadcasting cost.



Single-bit tree indexing

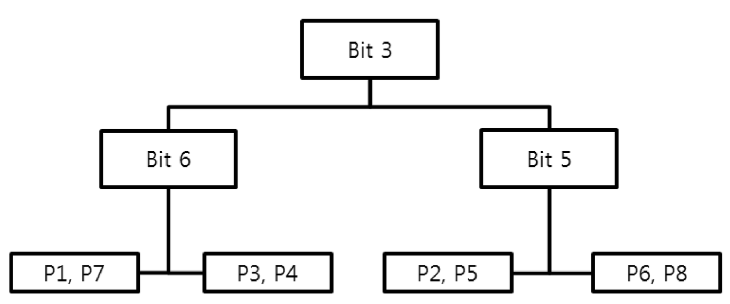
- The black squares represent the selected bits in bloom filter, and the gray squares show the bits that were selected at an ancestor. The red triangles represent the sub-trees.




| | B[1] | B[2] | B[3] | B[4] | B[5] | B[6] | B[7] | B[8] |
|---|---|---|---|---|---|---|---|---|
| A | 3 | 1 | 0 | 2 | 0 | 0 | 1 | 0 |
| B | 0 | 1 | 3 | 1 | 0 | 0 | 3 | 0 |
| C | 0 | 2 | 0 | 1 | 0 | 1 | 2 | 0 |
| D | 1 | 3 | 0 | 2 | 0 | 2 | 0 | 2 |
| E | 1 | 0 | 2 | 4 | 0 | 1 | 3 | 1 |
| F | 2 | 4 | 1 | 0 | 3 | 0 | 0 | 3 |
| G | 1 | 2 | 0 | 2 | 0 | 0 | 2 | 2 |
| H | 0 | 0 | 1 | 3 | 2 | 3 | 1 | 4 |

DHT-based p2p network using chord protocol

- PDSs establish a DHT-based P2P network using chord protocol and each PDS's count bloom filter index for POI. In this example, initial PDS is PDS A




Single-bit tree construction

- After the single-bit tree is built, the PDS assigns its address to the single-bit tree's corresponding node

## 4. PDS grouping analysis

### Privacy

| | B[1] | B[2] | B[3] | B[4] | B[5] | B[6] | B[7] | B[8] | B[9] | B[10] |
|---|---|---|---|---|---|---|---|---|---|---|
| x1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| x2 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| x3 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| BF(S) | **1** | **0** | **1** | **1** | **0** | **1** | **0** | **1** | **1** | **0** |
| v1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| v2 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| v3 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |

- x1, x2, and x3 are the elements inserted into the bloom filter BF (S)
- v1, v2, and v3 are the hiding elements.
- Bloom filter returns the true for x1 element, but it is deniable because v1, v2, and v3 also set the x1's bit position in the bloom filter.

- Deniability: If the bloom filter returns a true, we can deny the existence of the data by the bloom filter's false-positive property

$$\gamma(BF(S)) \approx \left(1-\exp\left(-\frac{vk}{m\left(1-e^{-\frac{kn}{m}}\right)}\right)\right)^{k}$$

Deniability equation

### Broadcasting cost

- The number of PDSs assigned to a leaf node is reduced while the single-bit tree's depth increases

### Update cost

- When data stored in the PDS is changed, false-positive and false-negative errors may occur

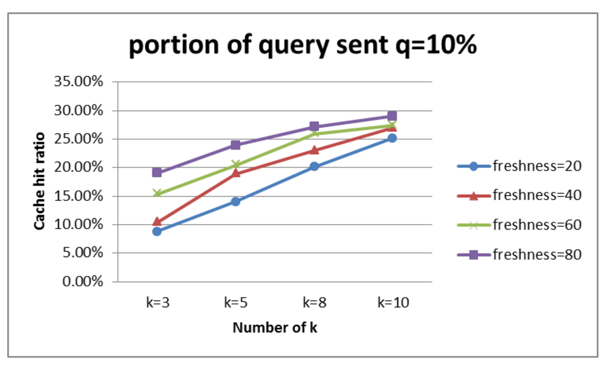
$$p=\sum_{i=0}^{k}\left[\left(\left(\frac{s}{m}\right)\times\frac{|X_1|}{m}\right)^{i}\times\left(\left(\frac{s}{m}\right)\times\frac{|X_c|}{m}\times\left(1-\left(1-\frac{1}{m}\right)^{kn}\right)\right)^{k-i}\right]$$

False-positive probability

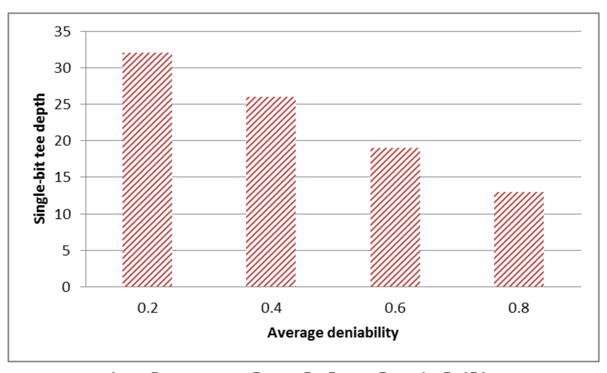
$$p=\frac{s}{m}\times\frac{|X_0|}{|X_0|+|X_c|\times\left(1-\left(1-\frac{1}{m}\right)^{kn}\right)}$$

False-negative probability

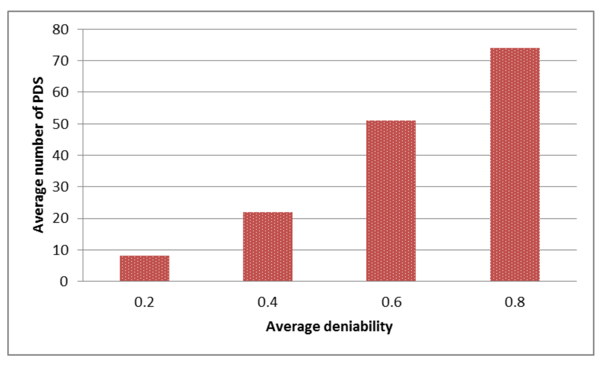
## 5. Experiment



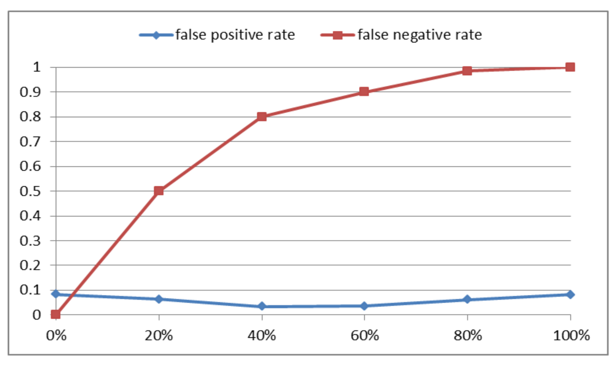

Collaborative caching performance




Single-tree depth by deniability




Average number of PDS by deniability




False positive and negative rate by single-bit tree inconsistency

- Simulation results show that the proposed technique achieves sufficient privacy protection while minimizing system performance degradation