FriendFinder: A Privacy Aware Grid Based Location Service

Ove Andersen Laurynas Siksnys Jeppe R. Thomsen

Aalborg University
Department of Computer Science
Selma Lagerløfs Vej 300, DK-9220 Aalborg Ø
Denmark

December 8, 2008

Introduction

- Project about a friend finding location based service (LBS).
- Notify a mobile user when any of his friends are within a certain distance.
- Dealing with location privacy issue.
- The server is said to be untrustworthy.
- Users do not want the LBS to know anything about their positions.
- Issuing continuous queries on private data.

Motivation

- How can location privacy be obtained if the LBS is untrusted?
- Can an anonymizer, which will be a single point of attack, be excessed?
- Is it possible to both retain sufficient privacy requirements and keep communication costs at a minimum?
- Not much work done on querying private data - most related work only considers private queries on public data.

Anonymizers are not Necessary

Private Queries in Location Based Services: Anonymizers are not Necessary.

- Proposes a framework to support private location-dependent queries.
- Does not require an anonymizer, since it's a single point of attack.
- Uses cryptographic techniques to ensure privacy.
- Communication and CPU intense, because of encryption.

Buddy Tracking

Buddy tracking - efficient proximity detection among mobile friends.

- Proposes a centralized server and a peer-to-peer method for tracking friends.
- Using the strips on the peer-to-peer algorithm - ensures privacy and reduces communication cost.
- Using a quadtree algorithm for the centralized algorithm.
- Still lot of communication using the peer-to-peer algorithm.
- Centralized server algorithm.has a lot of overhead and is outperformed when lots of users has joined the service.
- Privacy is not discussed in this article.

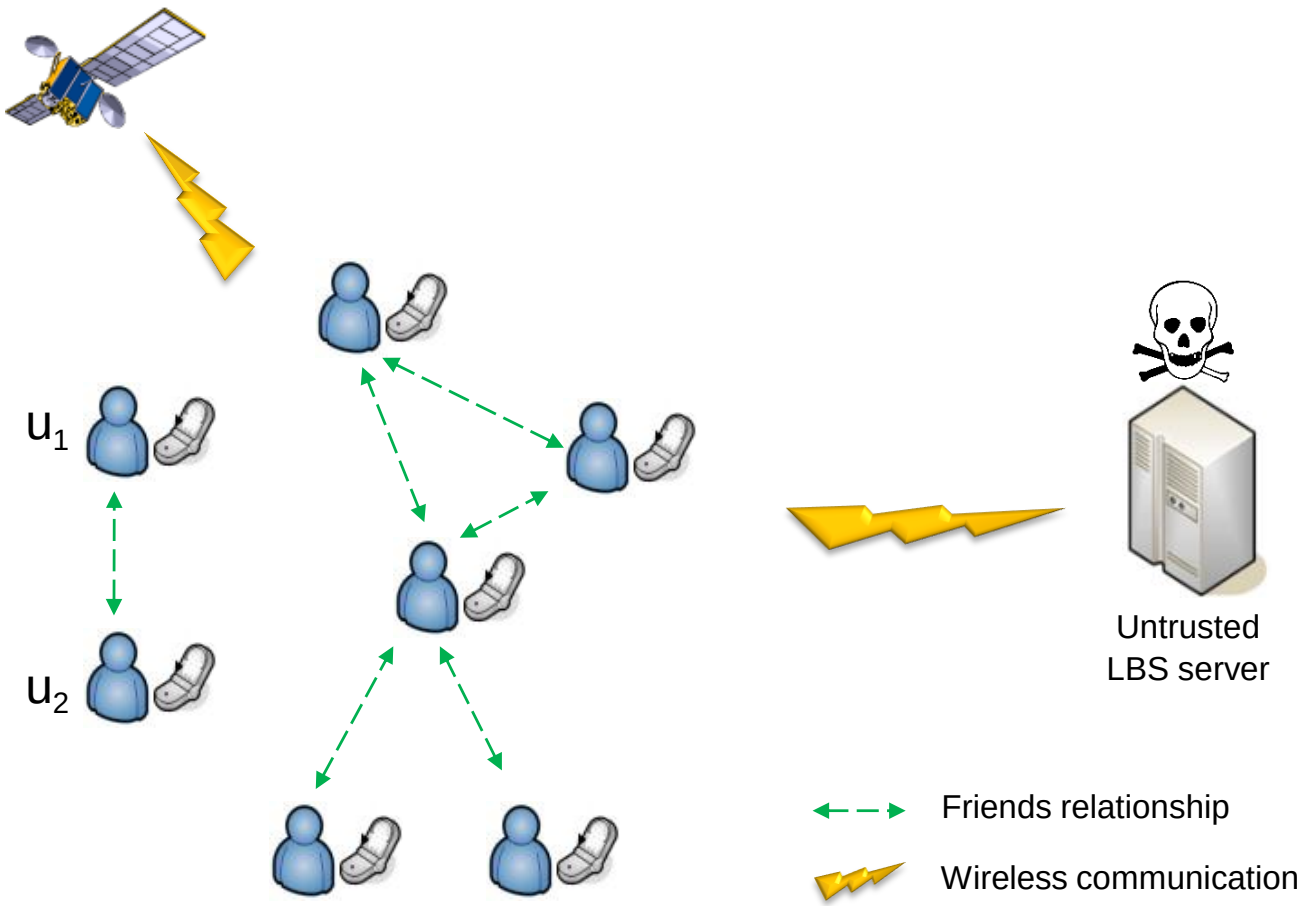
The contribution

- Agreed on the problem and its setting
- Develop a user location secure proximity detection approach
- Designed a Friend-finder service
- Develop a prototype
- Performed a prototype testing

The Problem Definition

- We want a solution for friend-finder LBS
- Solution must be strong against attempts to intercept exact locations of any user
- The solution must be practical

The Setting of the Problem

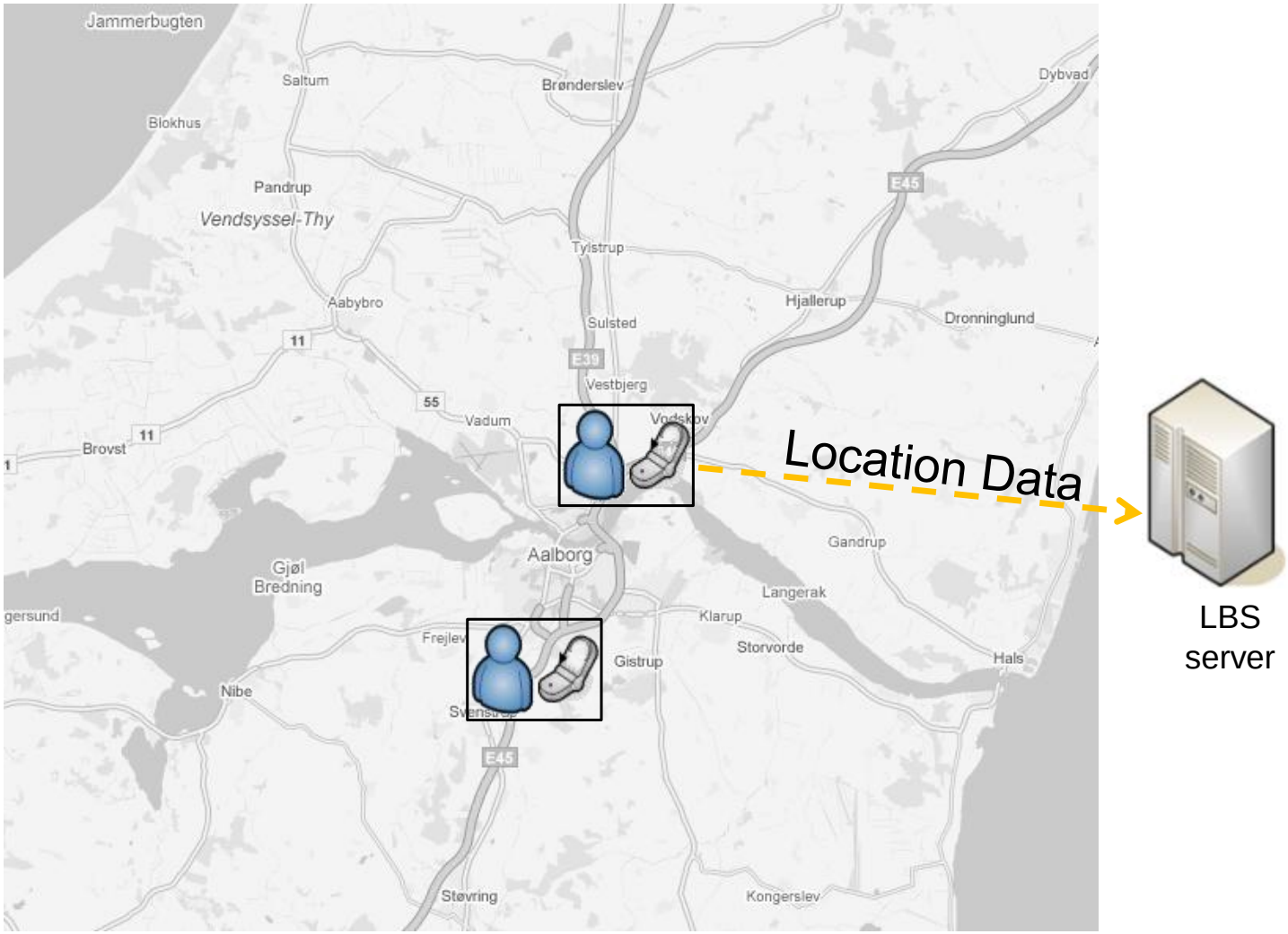


Example of the Proximity Detection

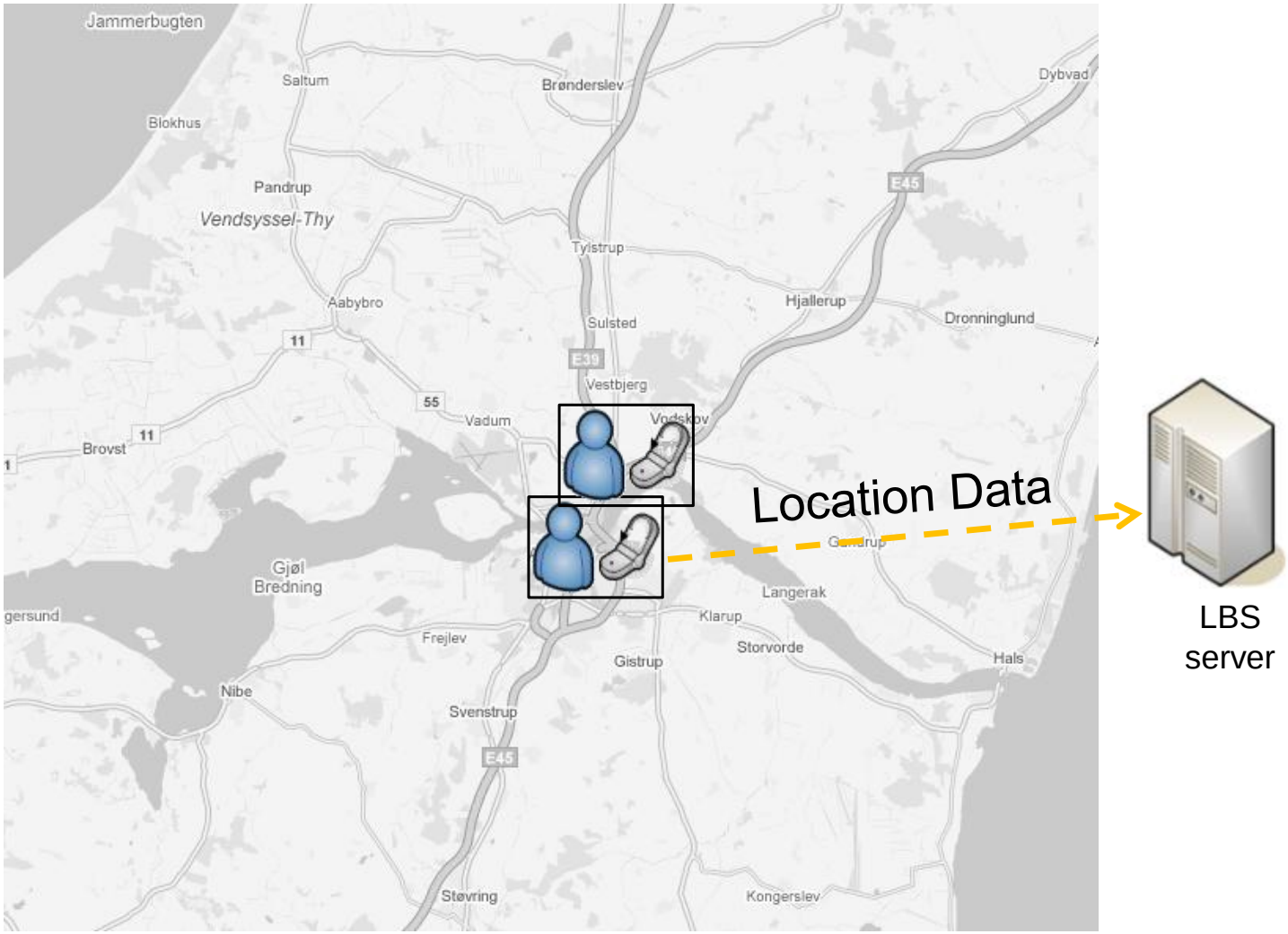
LBS server

Example of the Proximity Detection

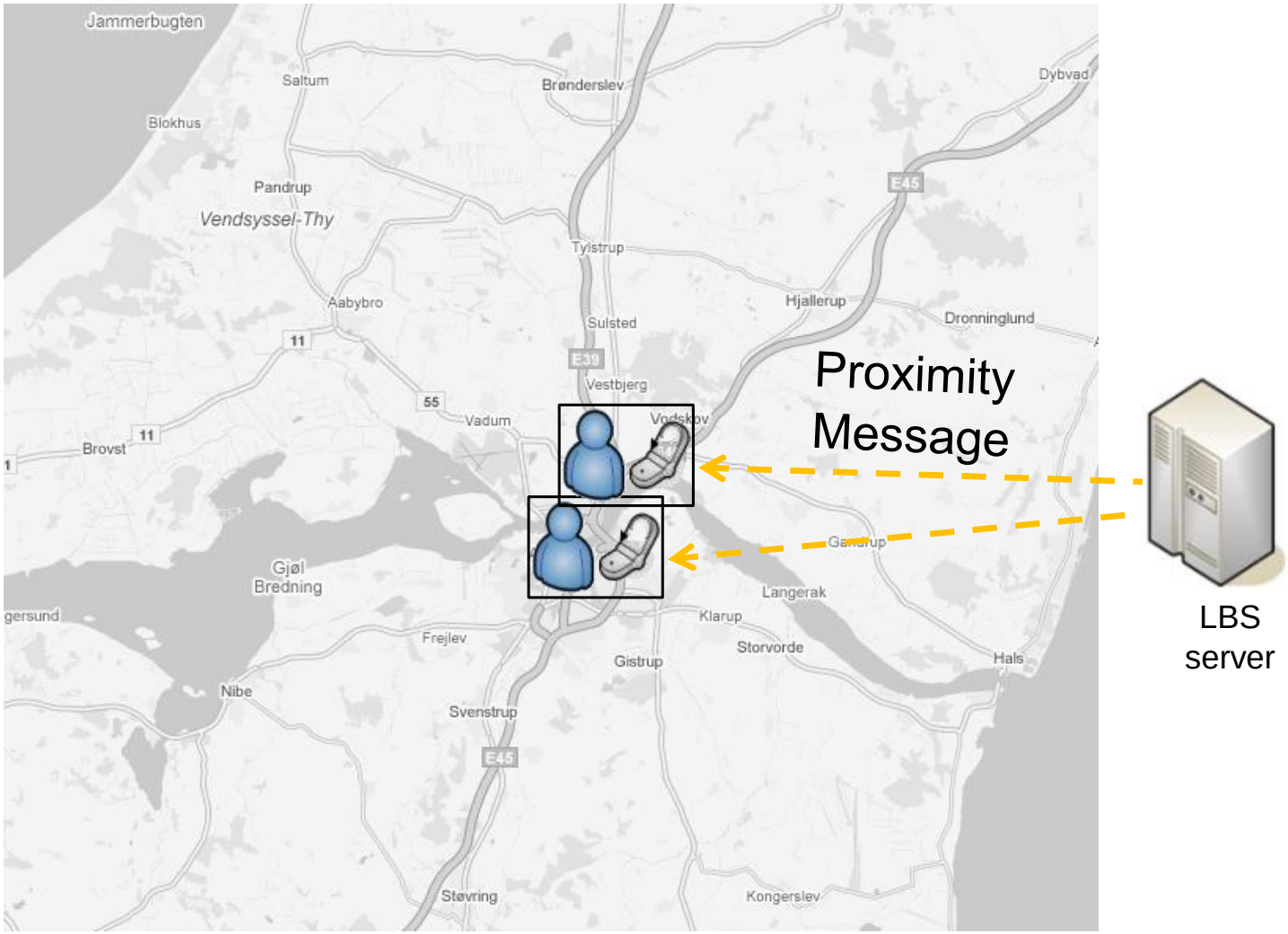
Example of the Proximity Detection



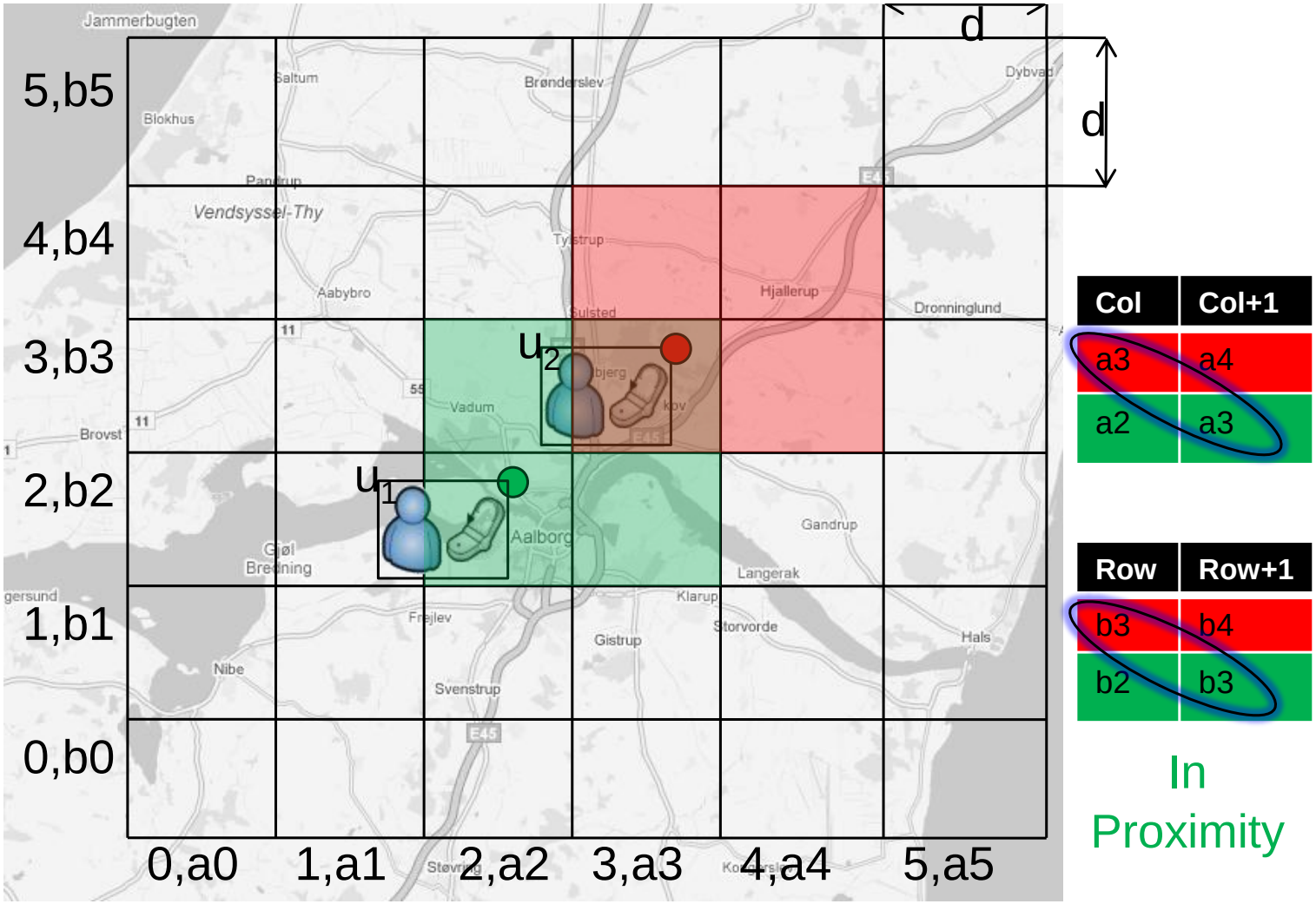
Example of the Proximity Detection



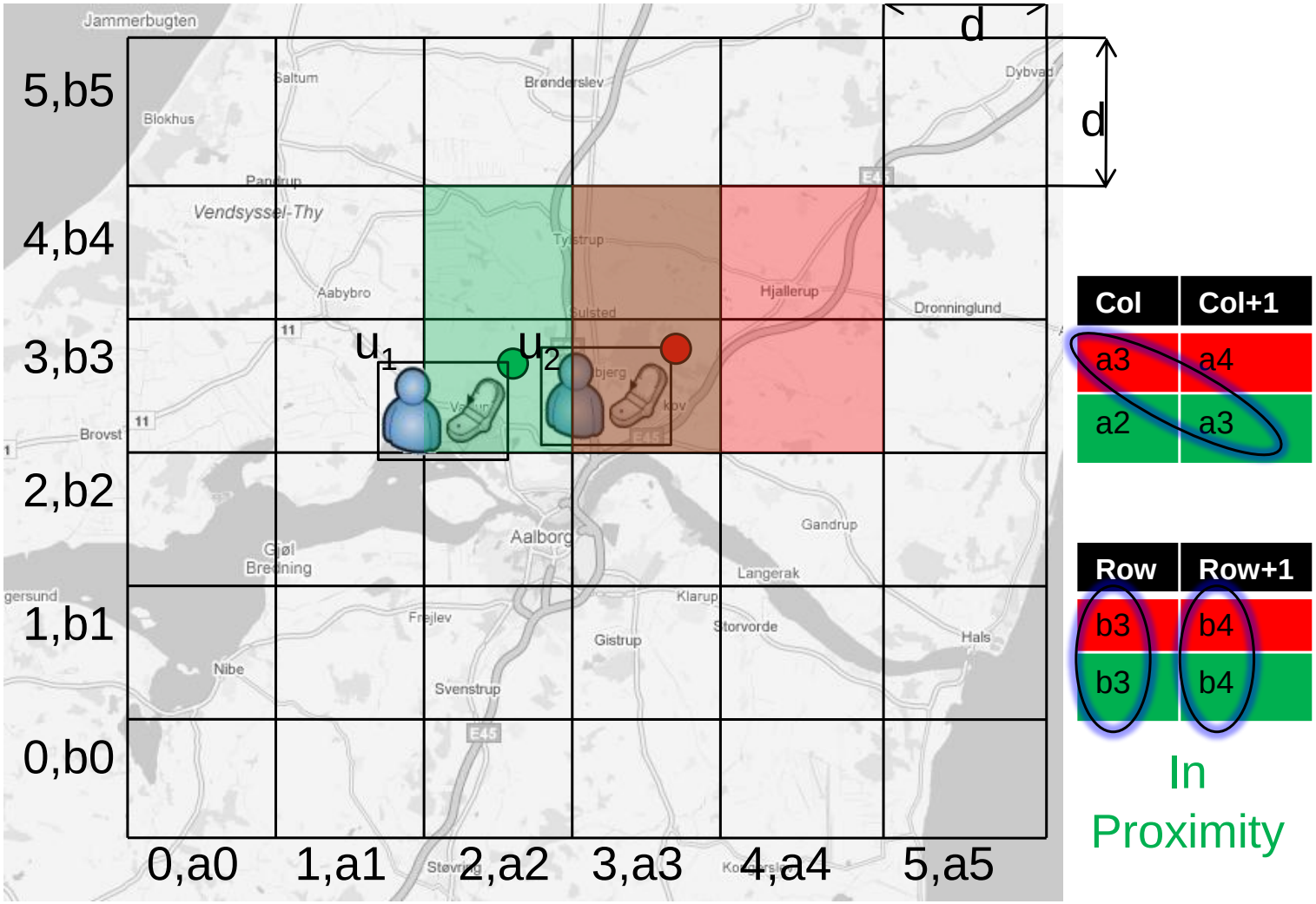
Example of the Proximity Detection



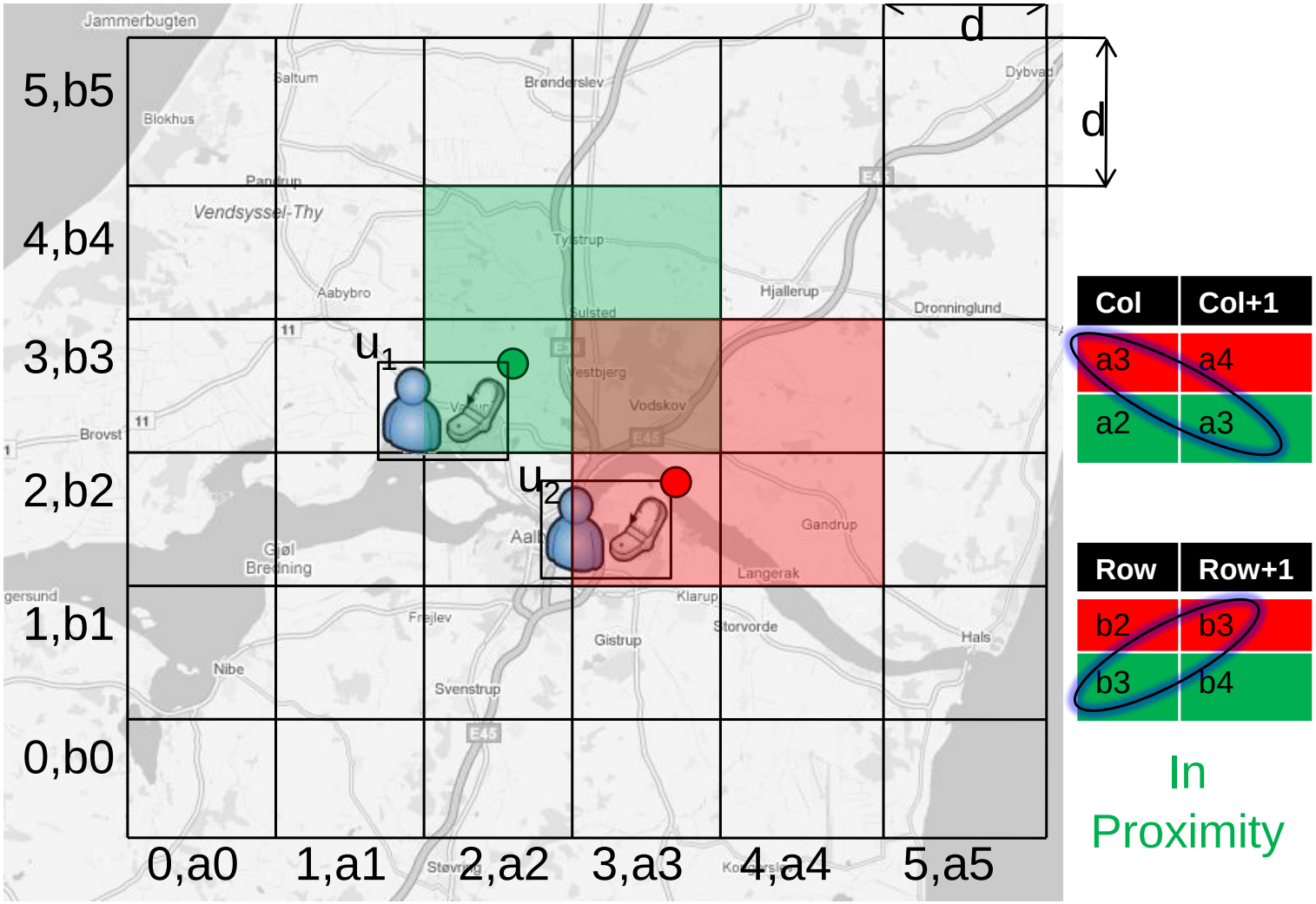
Privacy-aware Proximity Detection Idea



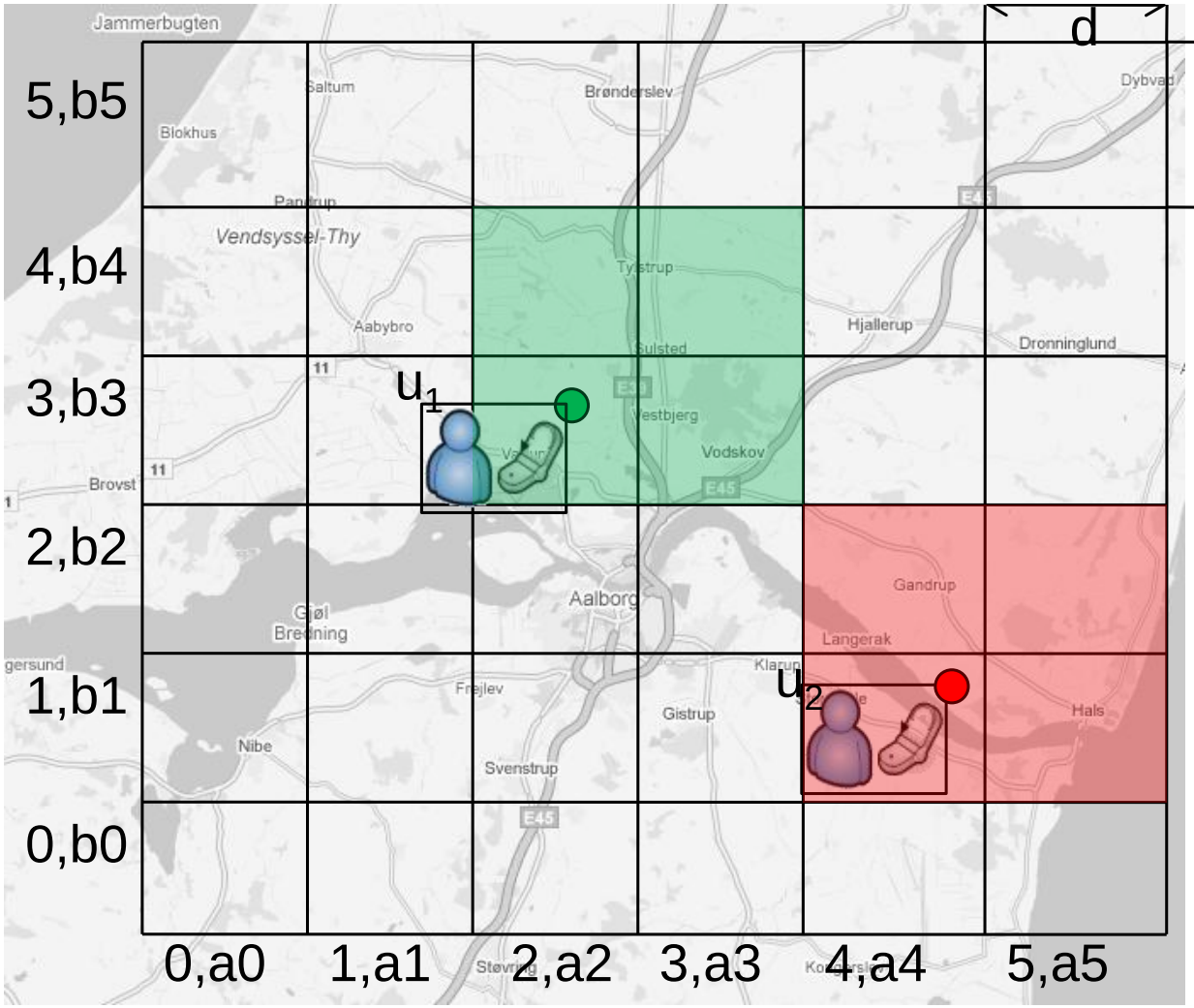
Privacy-aware Proximity Detection Idea



Privacy-aware Proximity Detection Idea



Privacy-aware Proximity Detection Idea



Col	Col+1
a3	a4
a2	a3

Row	Row+1
b1	b2
b3	b4

In Separation

Limitation and solutions of this approach

Limitations of this approach:

1. It is inefficient for big number of users
2. A proximity detection distance is fixed $(2d\sqrt{2})$

Solutions for problems:

1. Grouping of friends
2. Dynamic grid approach

Grouping of friends

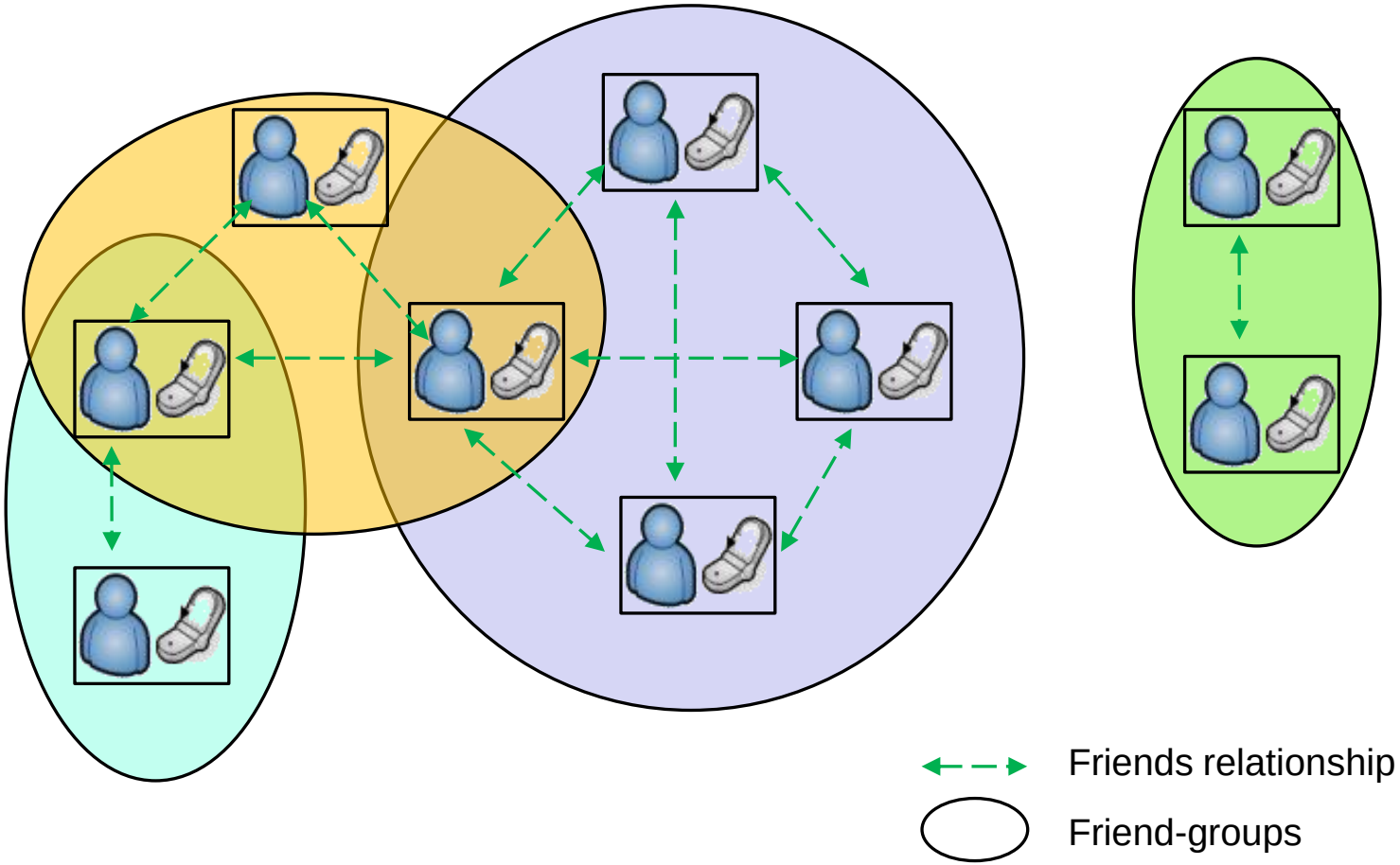
The solution:

- Users are grouped into friend-groups
- A single grid is assigned for every friend-group

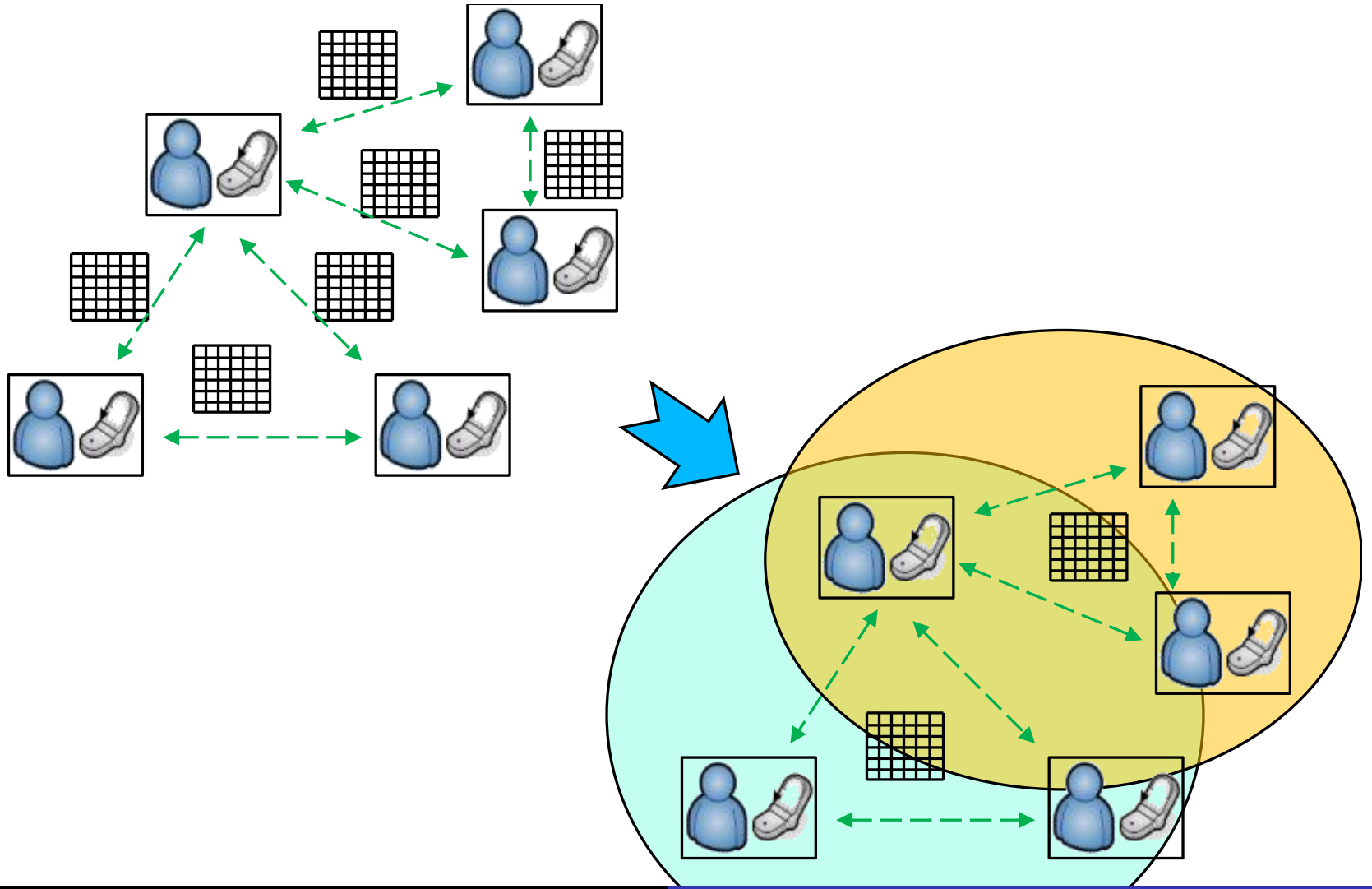
Consequences:

- When user location changes, less secret values must be delivered to LBS server

Grouping of friends



Grouping of friends



Dynamic Grid Approach

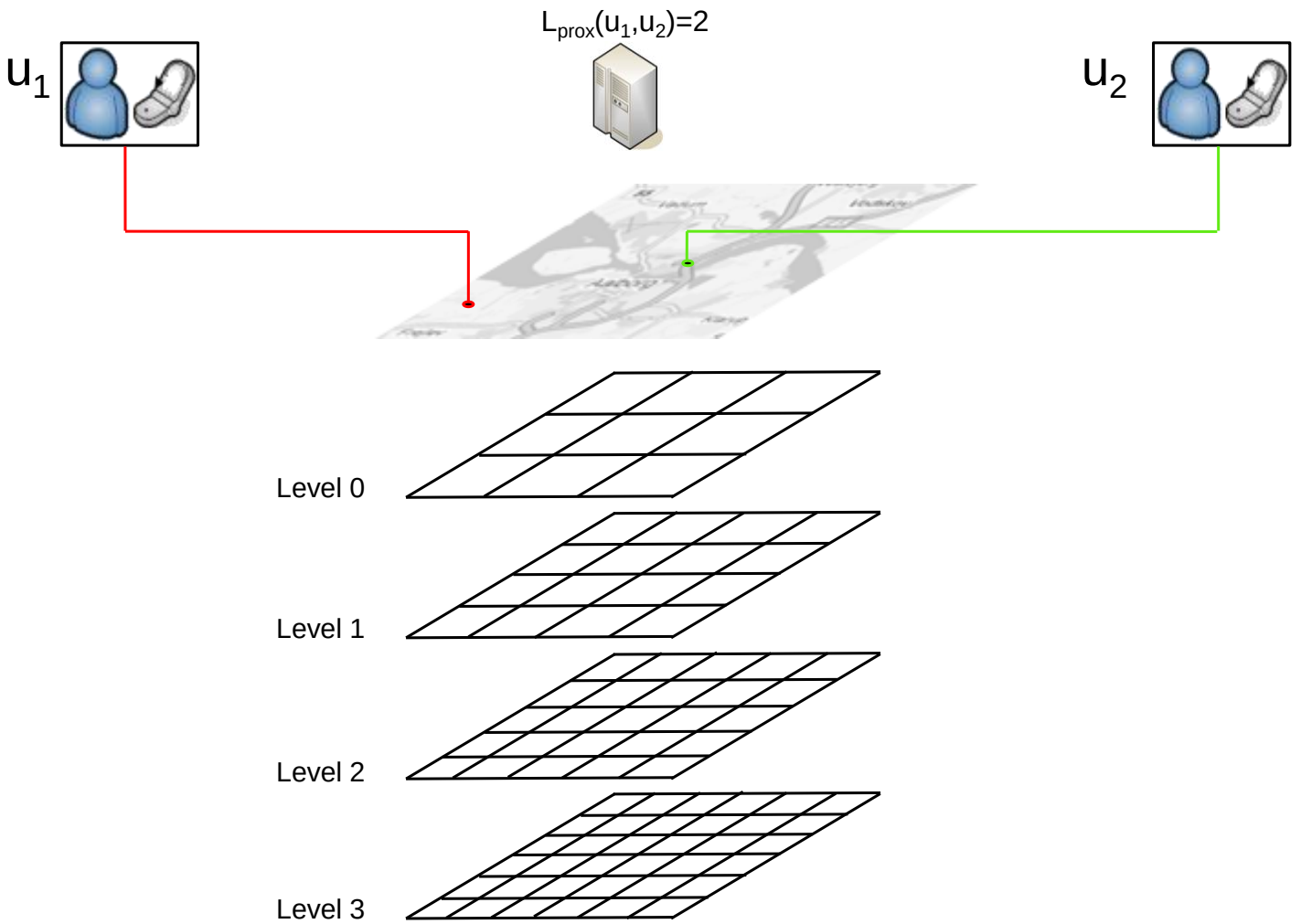
The solution:

- Assign a stack of grids with decreasing cell size for every friend-group
- Extend a basic proximity detection approach to support stack of grids

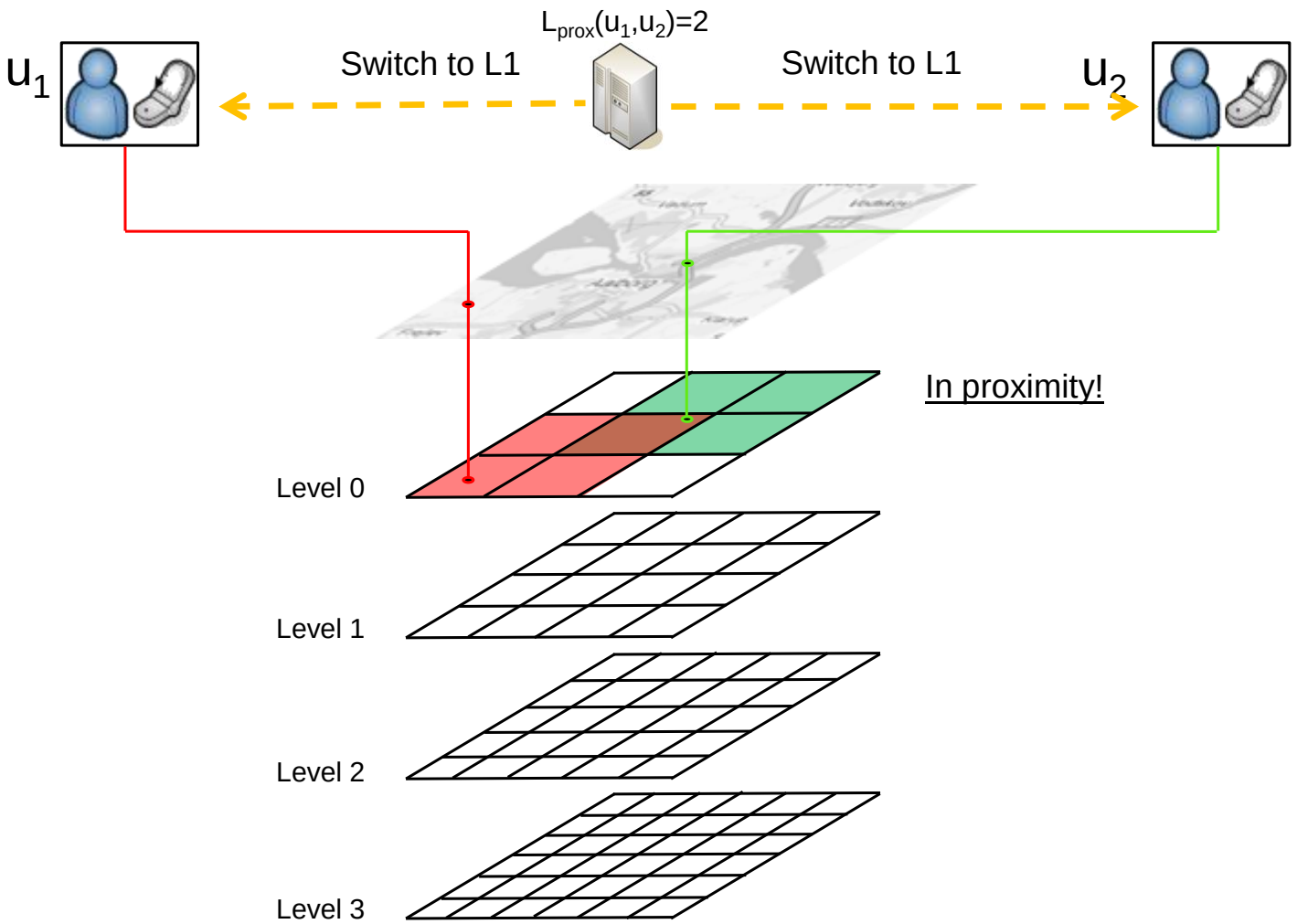
Consequences:

- Any pair of friends in friend-group will be able to choose a prefered proximity distance from a list

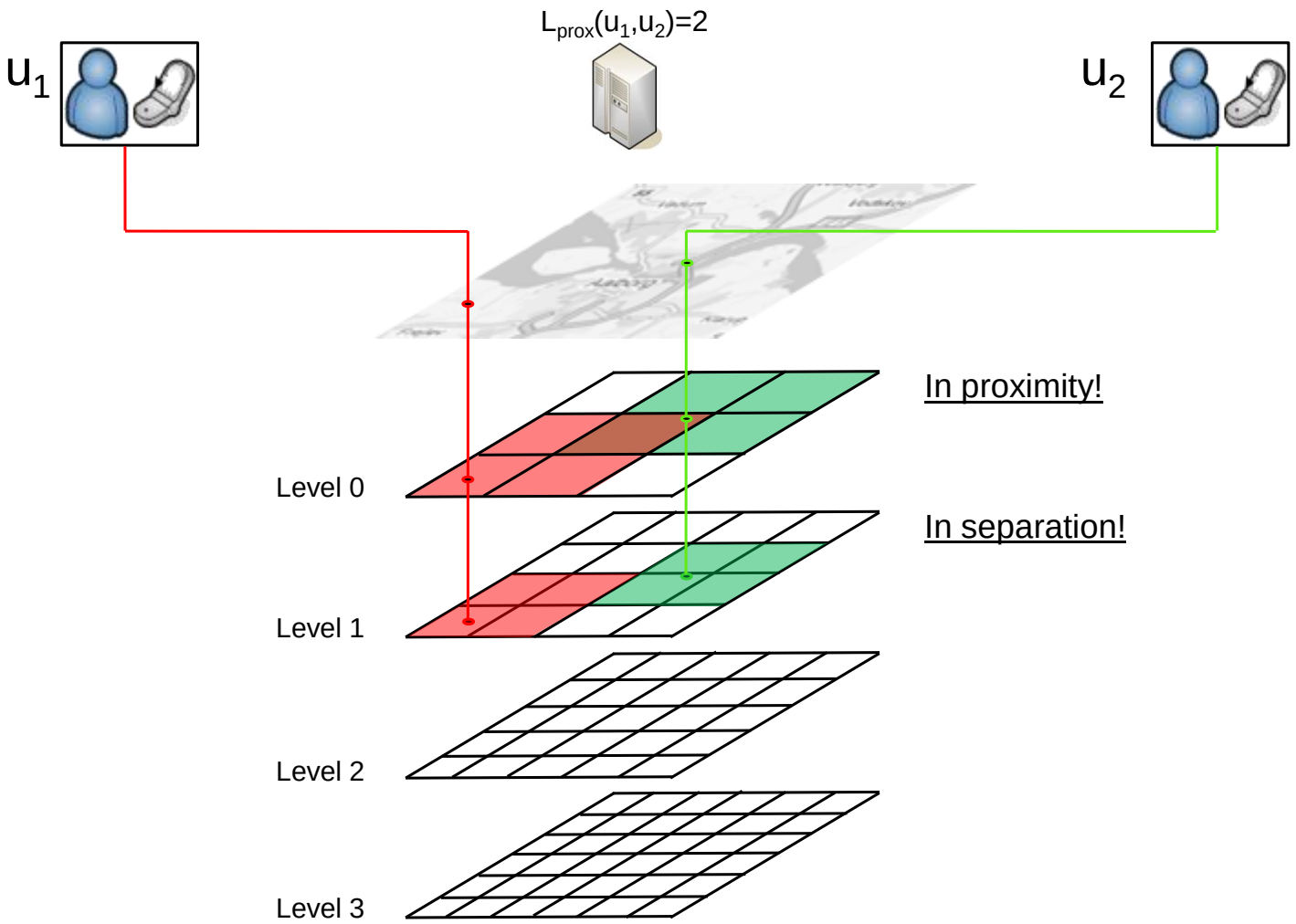
Dynamic Grid Approach



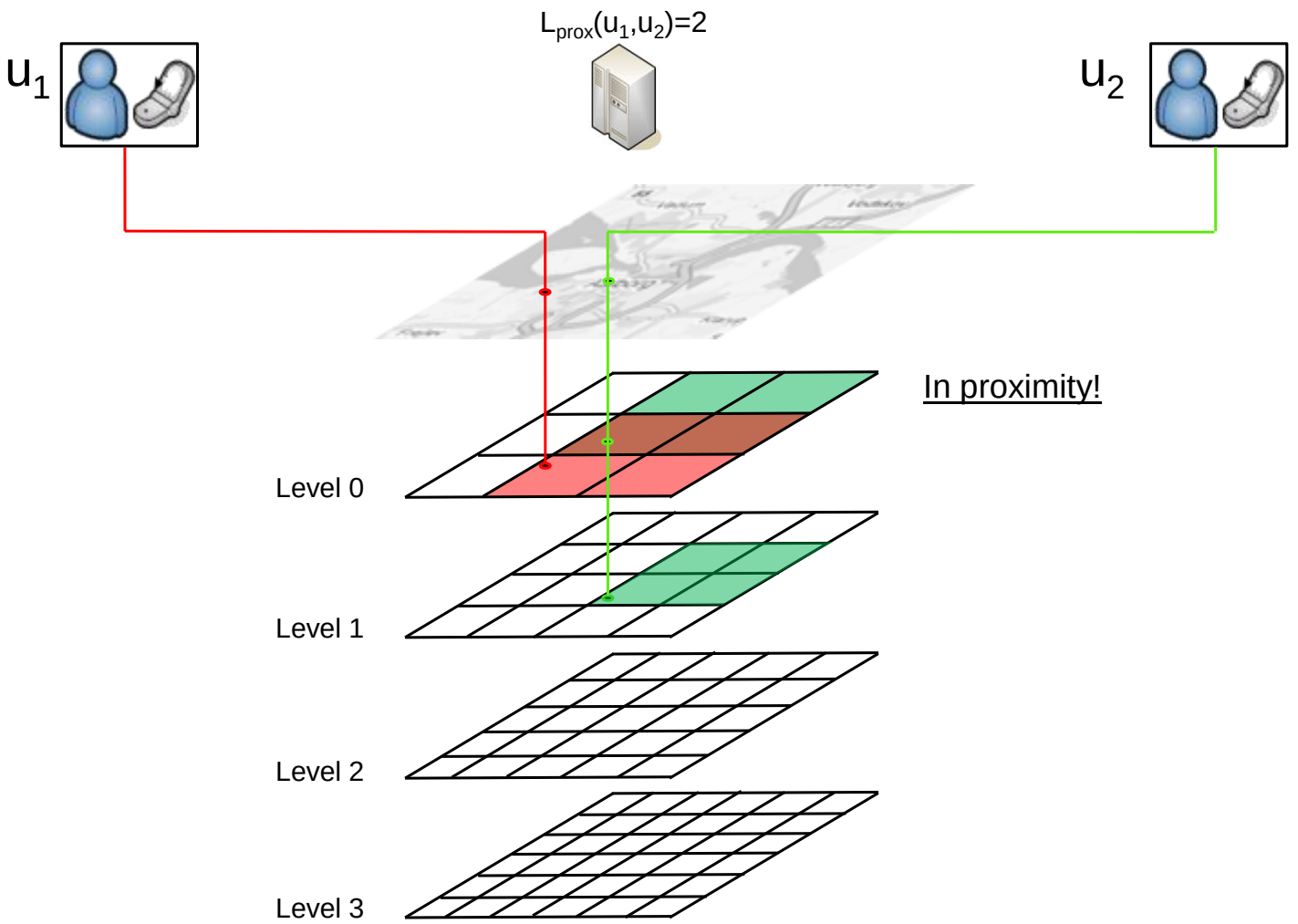
Dynamic Grid Approach



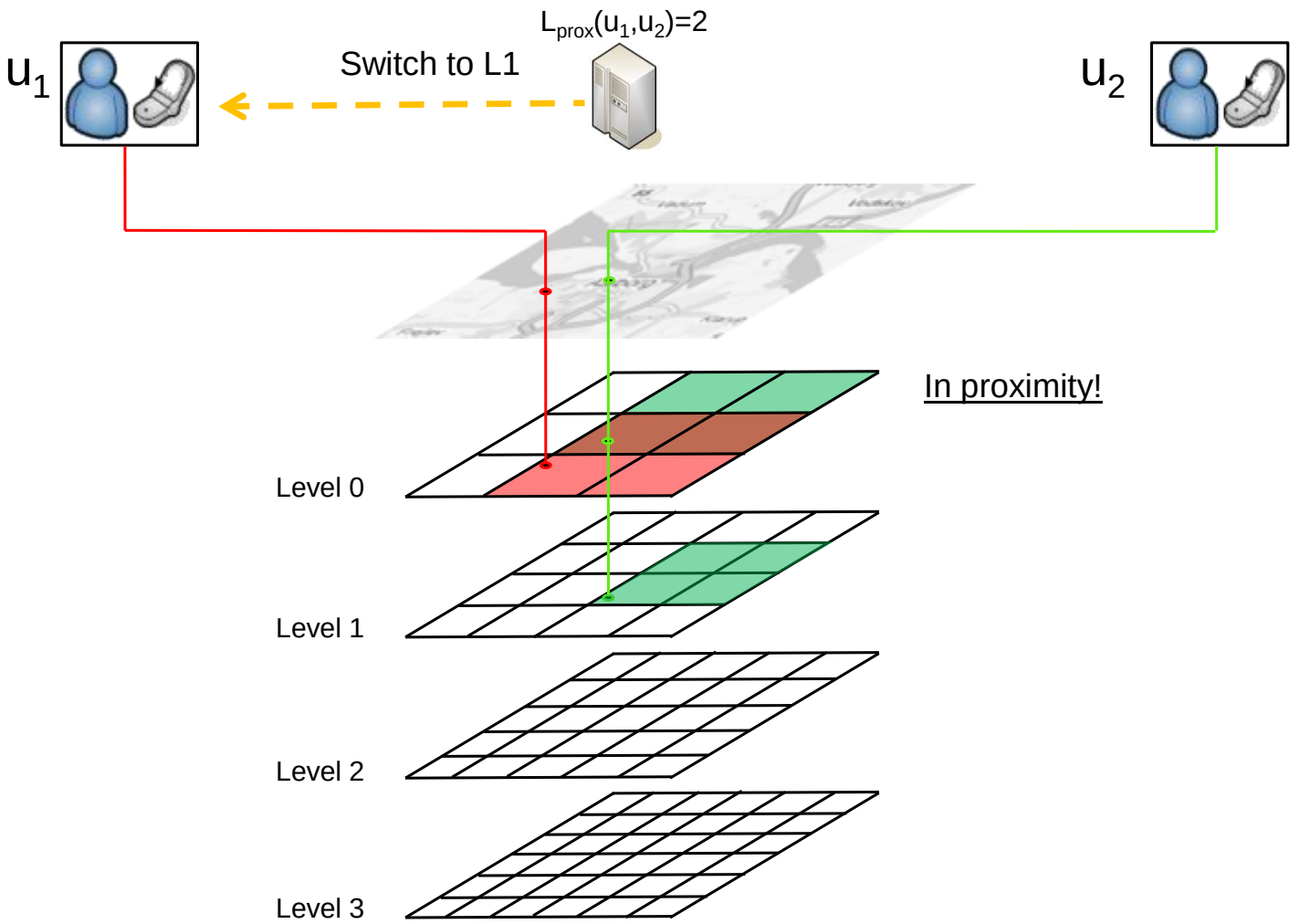
Dynamic Grid Approach



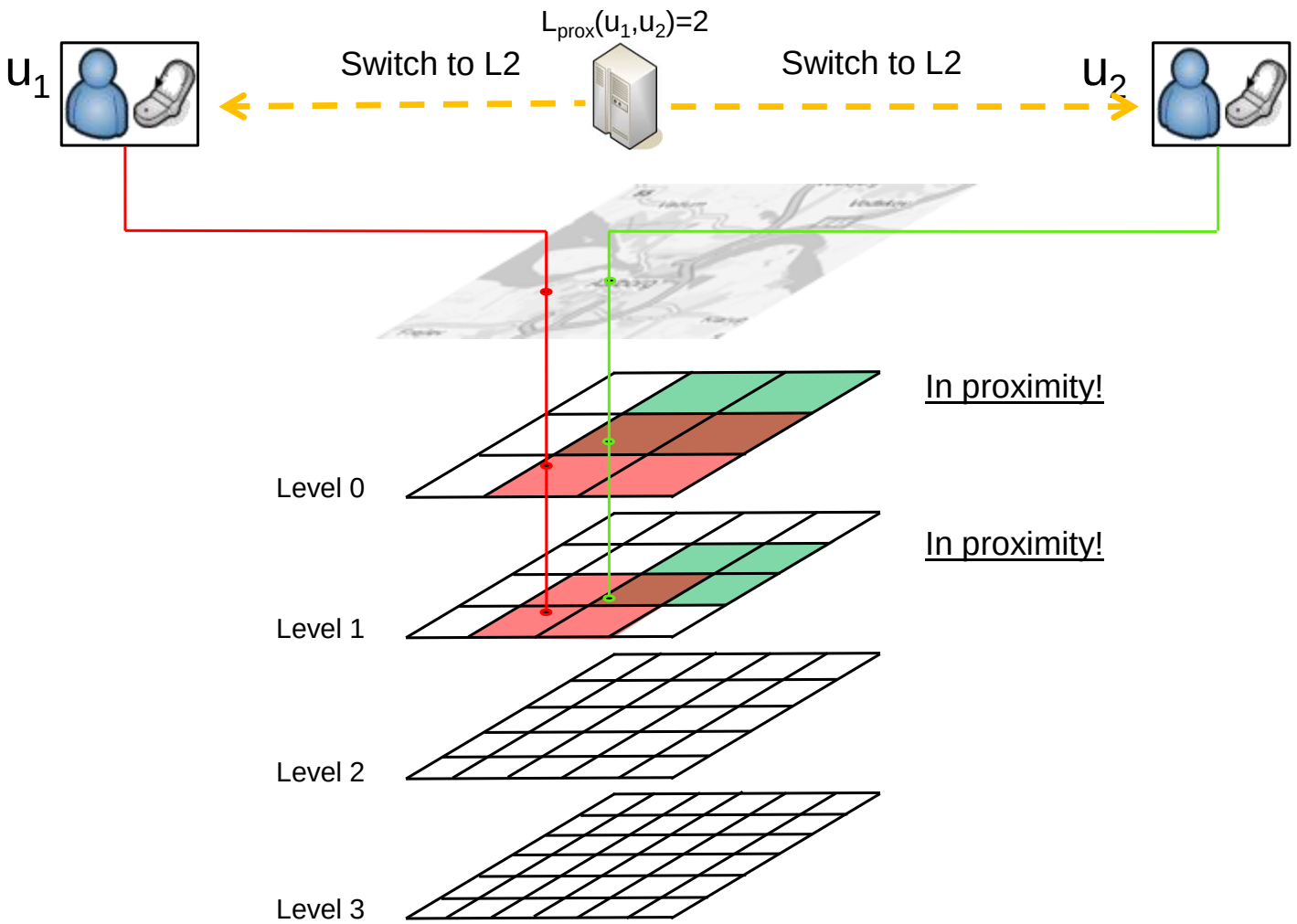
Dynamic Grid Approach



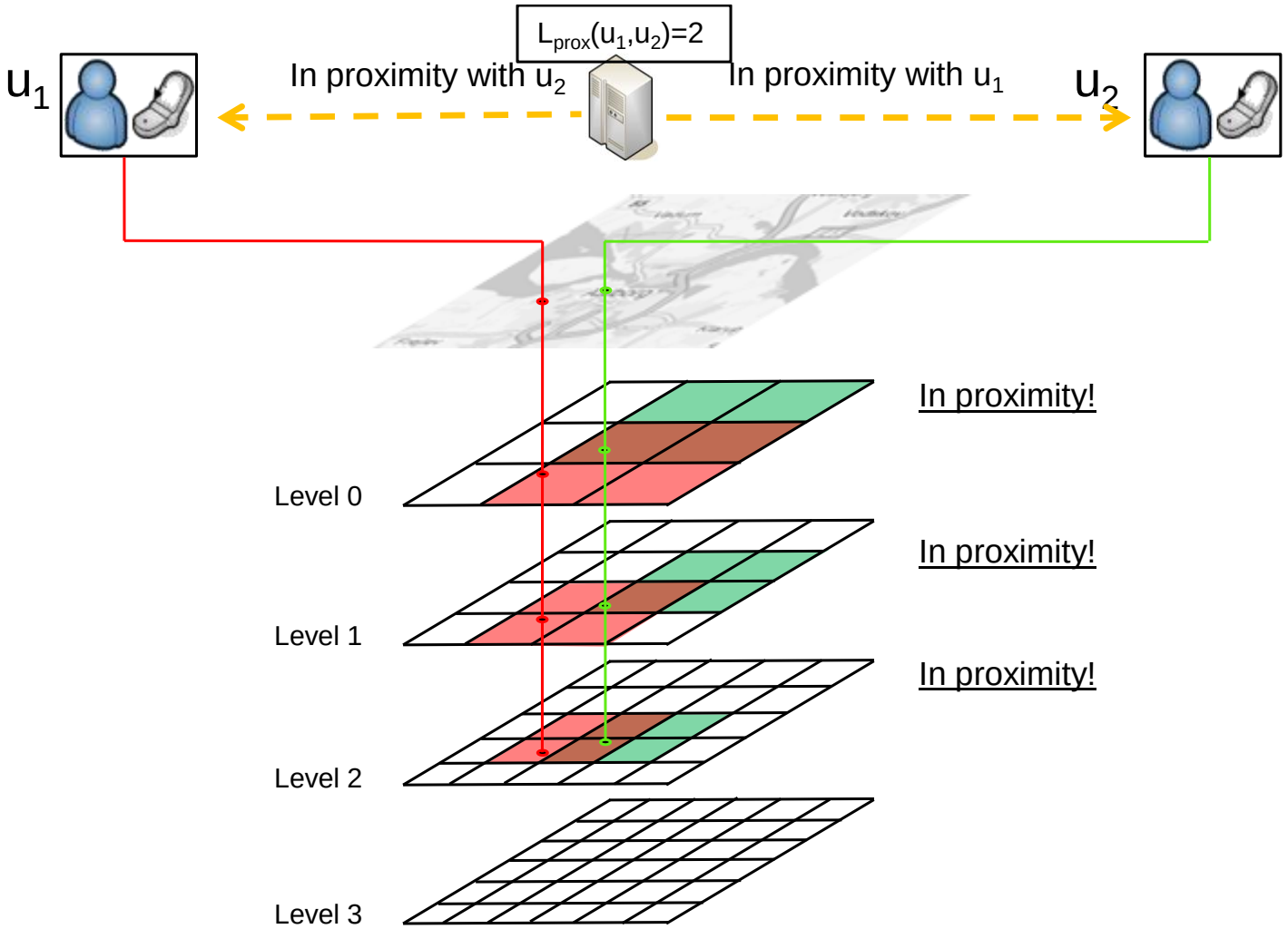
Dynamic Grid Approach



Dynamic Grid Approach



Dynamic Grid Approach



Friend-finder service designs

Two Friend-finder service designs:

- Basic Design
  - Uses basic proximity detection approach
- Dynamic Grid Design
  - Uses dynamic grid approach

Features:

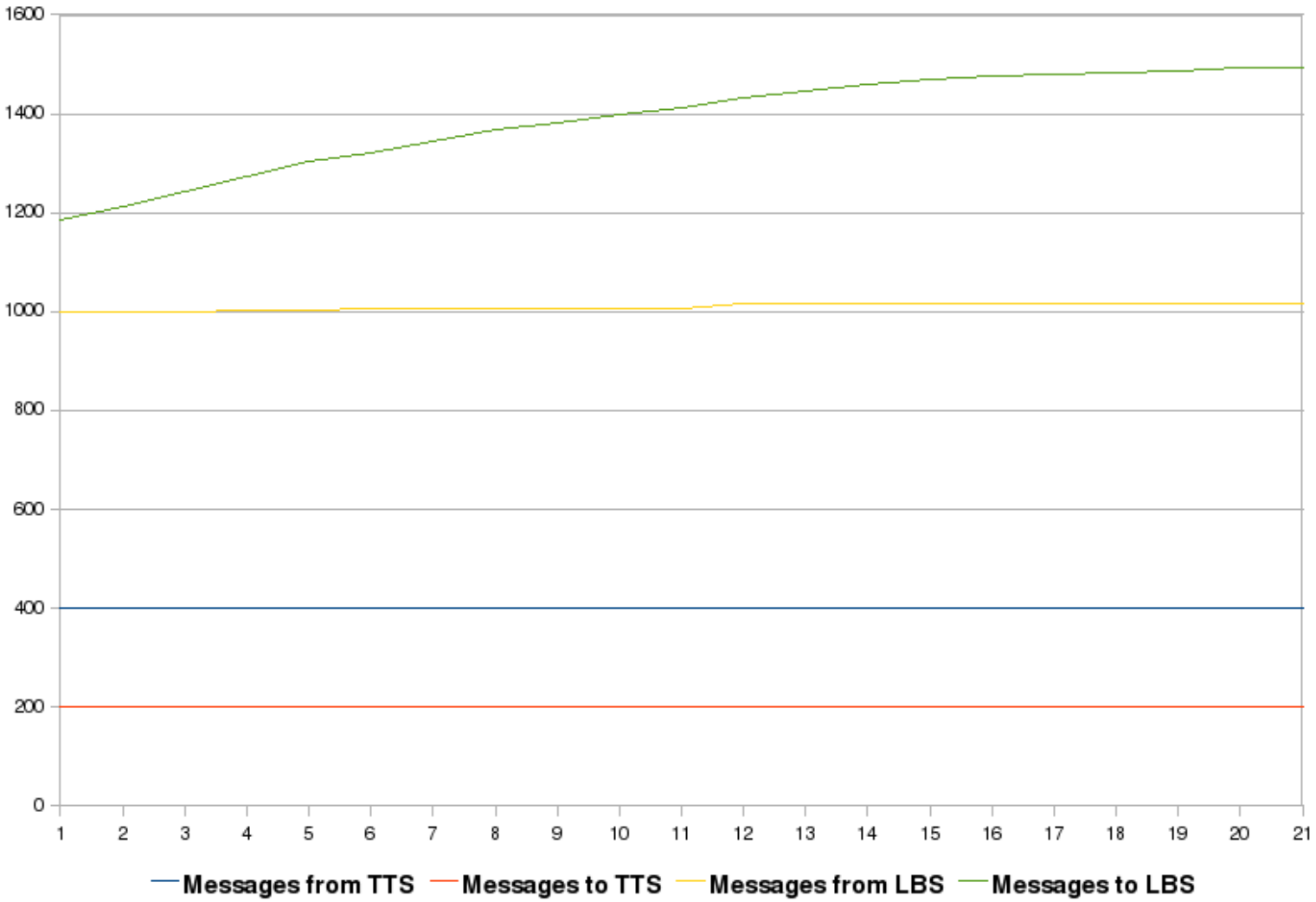
- Employs users grouping into friend-groups
- Assumes an existence of trusted 3rd party server

Testing Methodology

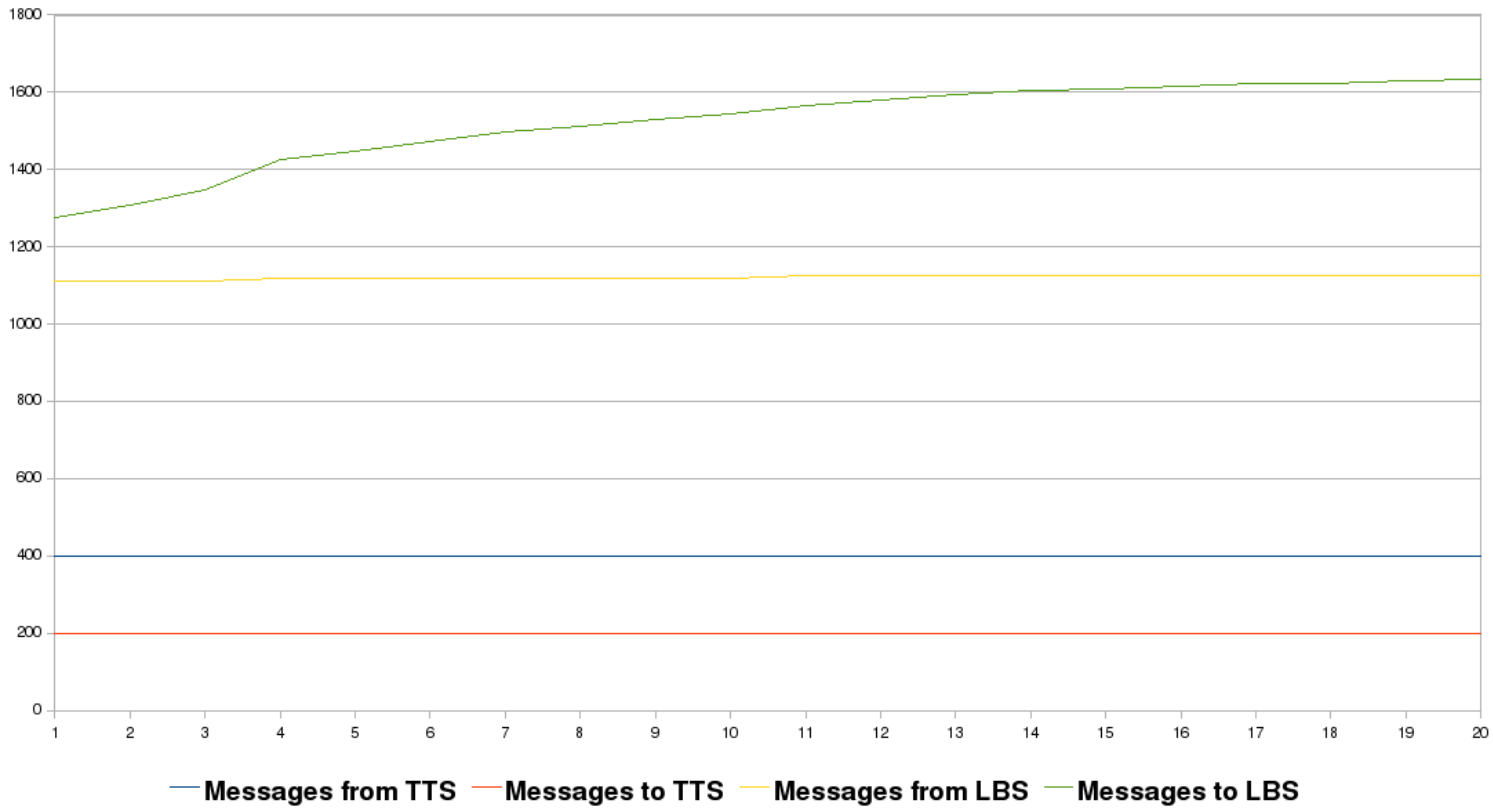
System Test Settings

- 200 users.
- 20 Position Points per user.
- Each user only in one group.
- Proximity level: 6
- Four data sets
  - 5 Users per Group (40 Groups).
  - 10 Users per Group (20 Groups).
  - 20 Users per Group (10 Groups).
  - 50 Users per Group (4 Groups).

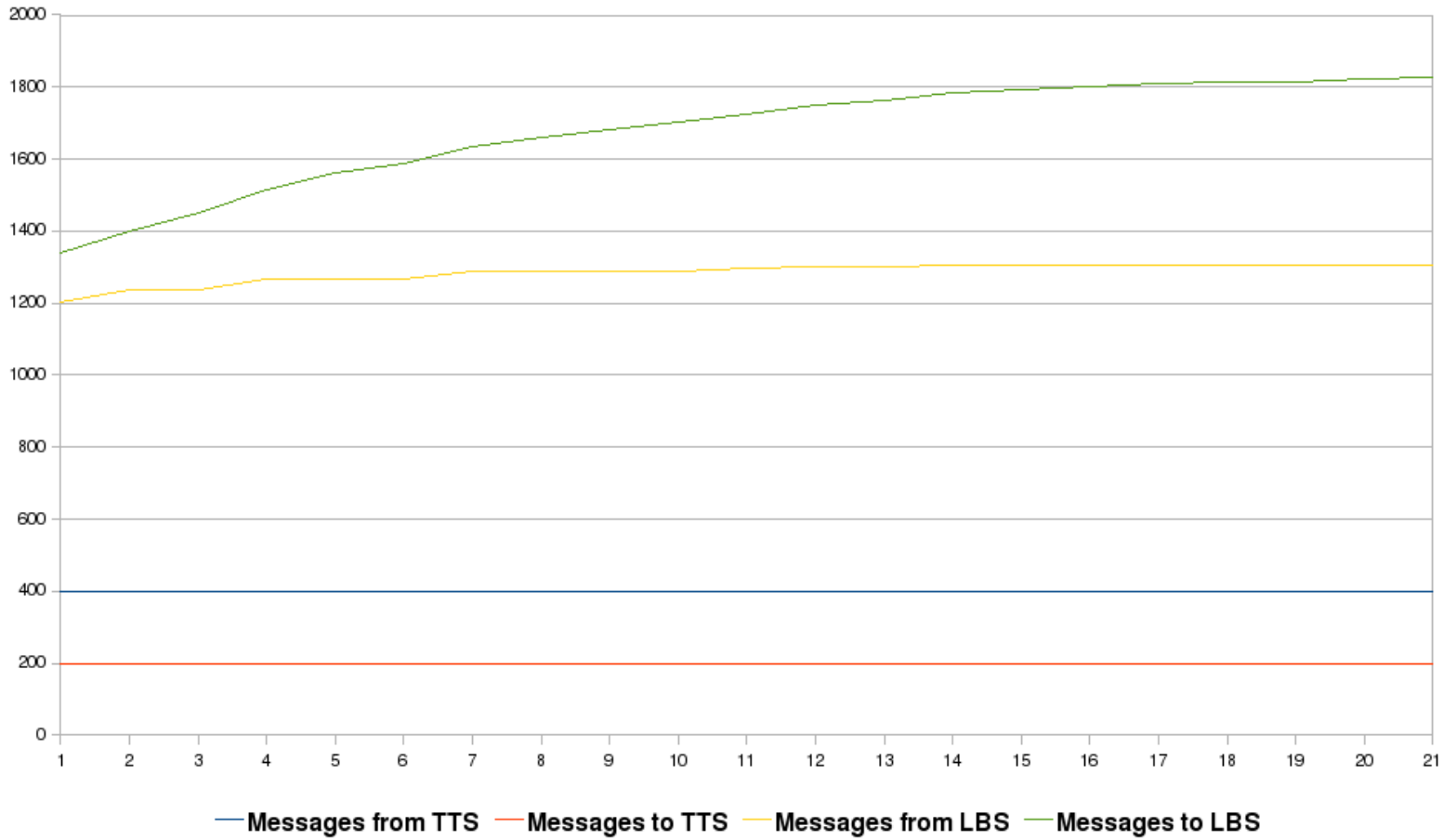
5 Users in each Group / 40 Groups



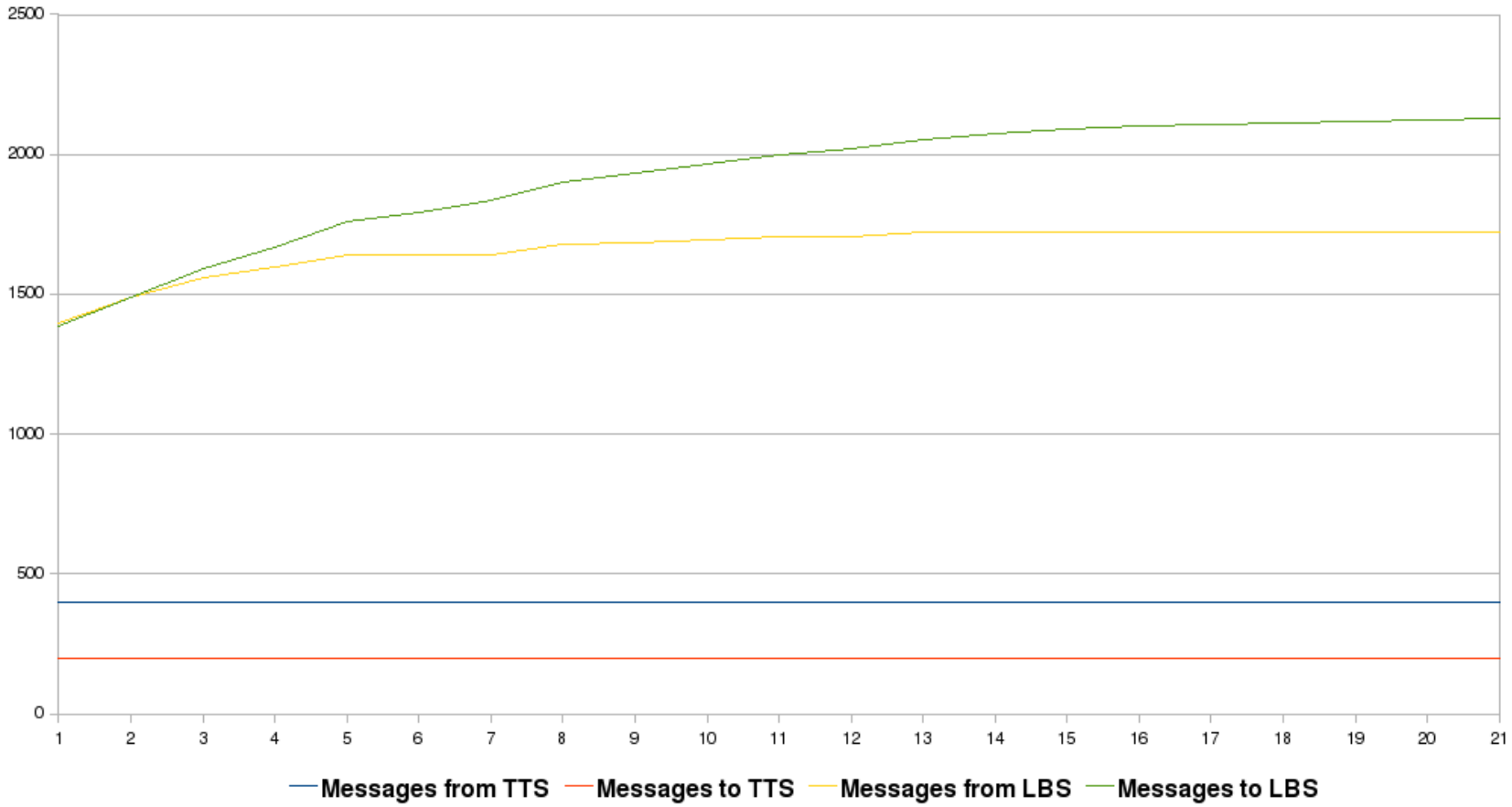
10 Users in each Group / 20 Groups



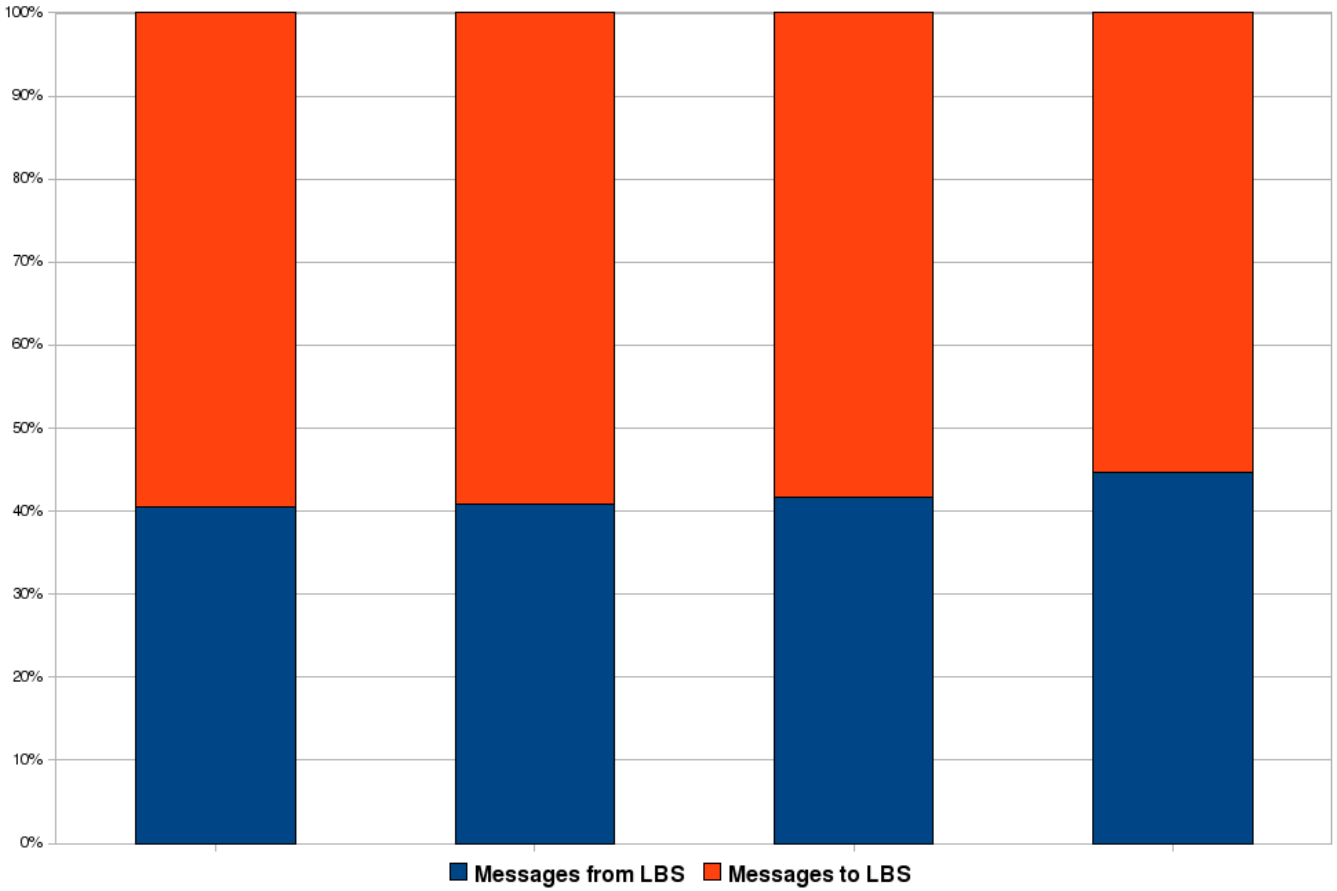
20 Users in each Group / 10 Groups



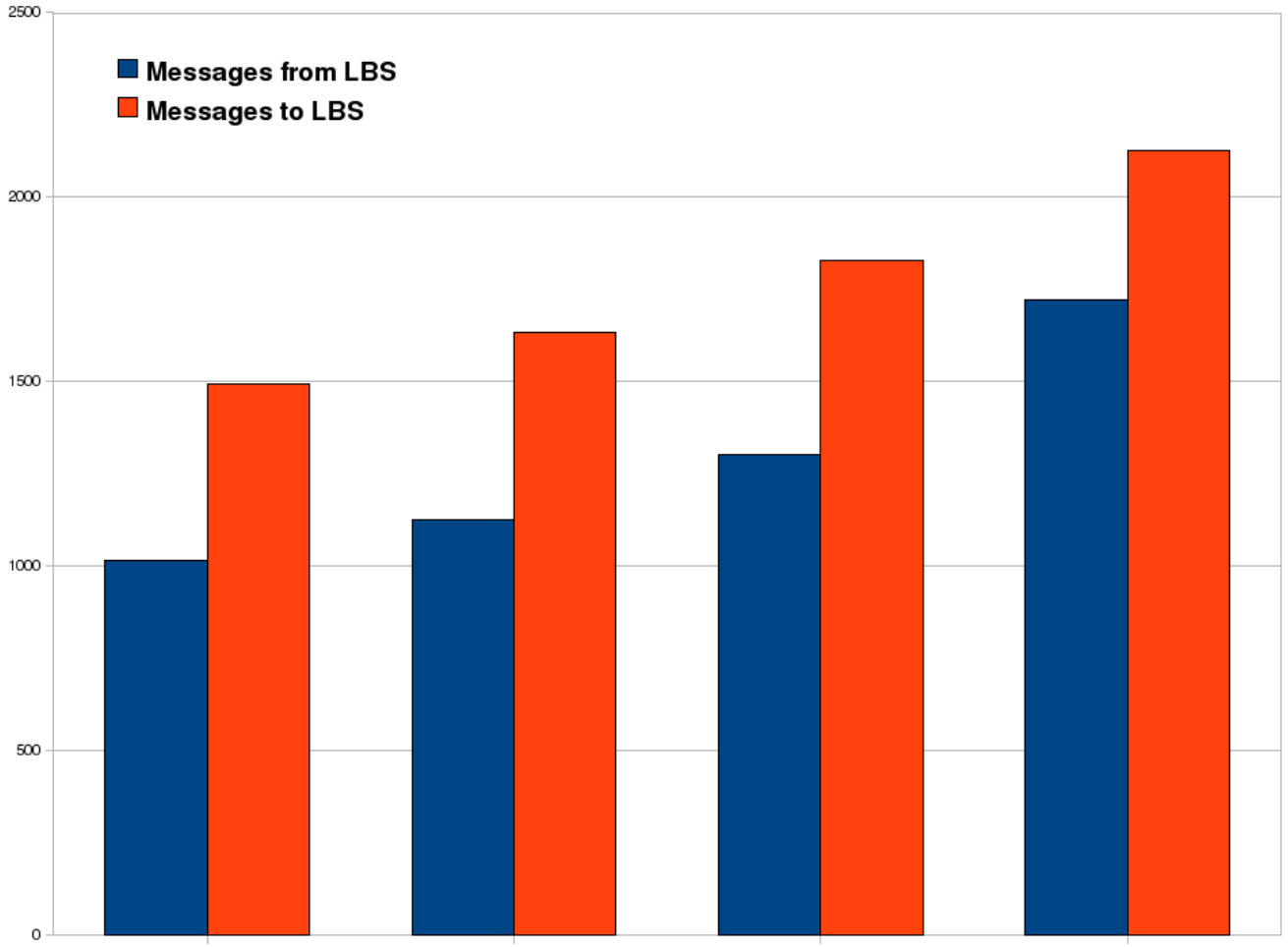
50 Users in each Group / 4 Groups



Total Messages, Percent-wise distribution



Total Messages



Conclusion

- Novel approach.
- System has a very low running cost in terms of messages
- Very resilient to attacks.

Future Work

Possible extensions

1. Different grid cell shapes.

Future Work

Possible extensions

1. Different grid cell shapes.
2. Notification of other users getting in proximity.

Future Work

Possible extensions

1. Different grid cell shapes.
2. Notification of other users getting in proximity.
3. Notification of a user getting in proximity of a static object.

Future Work

Possible extensions

1. Different grid cell shapes.
2. Notification of other users getting in proximity.
3. Notification of a user getting in proximity of a static object.
4. Notification of large groups of friends in proximity.

Future Work

Possible extensions

1. Different grid cell shapes.
2. Notification of other users getting in proximity.
3. Notification of a user getting in proximity of a static object.
4. Notification of large groups of friends in proximity.

Other Directions

1. Analyze user-levels on Server to find congested areas.

Future Work

Possible extensions

1. Different grid cell shapes.
2. Notification of other users getting in proximity.
3. Notification of a user getting in proximity of a static object.
4. Notification of large groups of friends in proximity.

Other Directions

1. Analyze user-levels on Server to find congested areas.
2. History analysis of user grid patterns, e.g to find carpooling opportunities.

Future Work

Possible extensions

1. Different grid cell shapes.
2. Notification of other users getting in proximity.
3. Notification of a user getting in proximity of a static object.
4. Notification of large groups of friends in proximity.

Other Directions

1. Analyze user-levels on Server to find congested areas.
2. History analysis of user grid patterns, e.g to find carpooling opportunities.
3. Make the idea fully or partially peer-to-peer

Questions

- Interesting future work?
- Would you use such a system?
- How can a group of friends, through an untrusted server, exchange private information (e.g. an encryption key)