Proactive Replica Placement Using Mobility Prediction

Julien Gossa[1], *Andreas G.K. Janecek*[2], Karin A. Hummel[2], Wilfried N. Gansterer[2], Jean-Marc Pierson[3]

[1] LSIIT, Universit de Strasbourg, France

[2] Faculty of Computer Science, University of Vienna, Austria

[3] IRIT Lab., University of Toulouse, France

January 22, 2008

Acknowledgements

- This work was supported by The AMADEUS Project
- PAI - Asso Egide - Partenariat Hubert Curien (PHC)
- Founded by OEAD – Austrian Exchange Service

- Two teams:
  - Distributed and Multimedia Systems, Vienna
  - Ex Systemes d'Information Pervasifs, INSA Lyon

Outline

1. Replica Placement Algorithm (FReDi)
   - Reactive FReDI
   - Adding Prediction to FReDI
2. Mobility Prediction
3. Evaluation
4. Results
5. Conclusion

FReDi

FReDi - **F**lexible **Re**plica **Di**splacer

Flexible management system for the dynamic placement of content replicas [1] over a network of proxy-caches (PCs)

Distributed version of *DC-Tree*, an approximation algorithm for the *k-center* problem

Main goals: Optimize the replica placement in order to

- reduce the number of replicas and limit the load on servers and proxy caches (PCs)
- while keeping the best end-user QoS

[1] Note: we consider a *replica* a copy of a content managed by the proxy-caches network

Constraints

Network constraints

- PCs only know their direct neighbors
  (comparable to P2P environments)

⇒ lack of knowledge about network architecture

Proxy caches

- can *store* replicas
- can *delete* replicas
- can *share* replicas and messages with *direct* neighbors

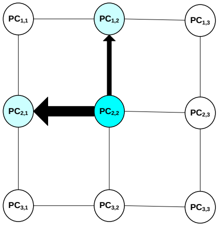
Attraction Vectors

Decisions for moving, duplicating and deleting replicas are supported by *“attraction vectors”* (AVs)

Each PC maintains one AV for each replica it stores

- each AV is composed of certain values (*attractions*) for each of the direct neighbors of the PC
- attractions represents the directional popularity of the replica toward a corresponding neighbor
- example: $AV(PC_{2,2})^t = (.2, .8, 0, 0)$

Example: AV for a replica [0.1em]held by $PC_{2,2}$



Attraction Vectors - Decisions

Three decisions:

- migration

  if an AV shows one single high attraction then the replica is migrated to the corresponding PC

- duplication

  if an AV shows several high attractions then the replica is duplicated to the corresponding PCs where the AVs exceed a given threshold

- deletion

  if an AV is null then the replica is of no use and is deleted

Attraction Vectors - Mechanisms

Two mechanisms:

- *AV*Update

  implements a distributed protocol designed to *adapt the attractions* of the relevant AV after each single request

  (using information of direct neighbors)

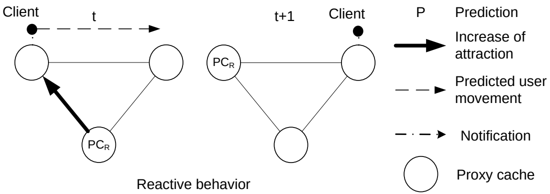
- *AV*Maintenance

  implements a local routine running on each PC designed to take into account the three decisions mentioned before

  (using only local AV information)

⇒ completely distributed – no centralized decisions

Reactive Behavior



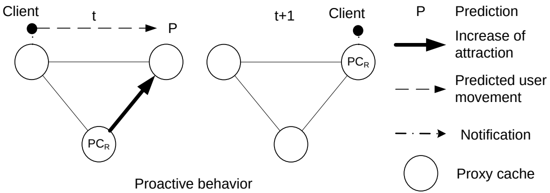
Reactive behavior

Problem:

Without considering future movement, replicas will basically follow *after* the clients have moved

PC_R indicates the PC holding the replica

Proactive Behavior



Proactive behavior

Idea:

Apply (mobility) prediction in order to make FReDi proactive

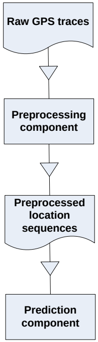
Update AVs *prior* to movement into the direction of the predicted future location of mobile client

Outline

Components

Preprocessing component

- convert raw history data traces into format used by the prediction component
- two levels of granularity (accuracy)
  1. ***granularity of time (gt)***
     determines the granularity of the time resolution underlying the location sequences
  2. ***granularity of position (gp)***
     determines the granularity of the position (i.e., accuracy of latitude and longitude)



Prediction component

- actual prediction implemented as comparison strategy

Granularity of Time

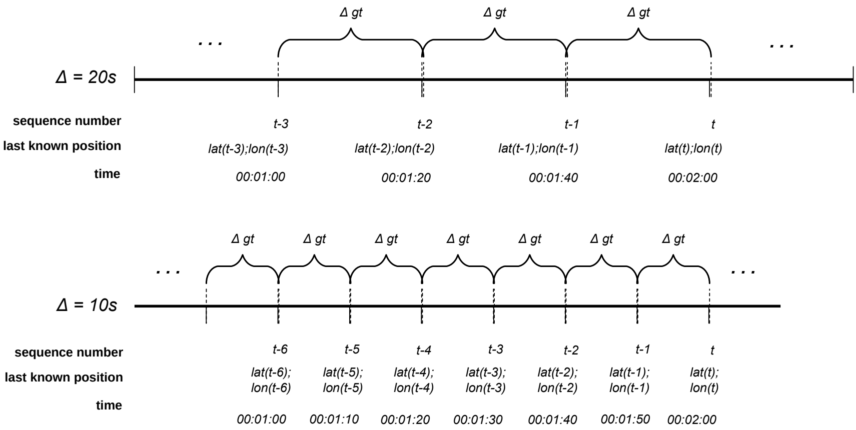
Discretize location sequences

- locations of raw GPS data are given in varying time intervals
- define basic time interval Δgt
- convert raw data traces into *discrete* location sequences containing the location of each mobile entity every Δgt seconds
- Δgt can be adjusted to the application

Filter out traces of non-moving mobile clients

- if mobile entity is not moving for a *minimal standing time* we start a new trace sequence for the given entity
- e.g., taxi stand, traffic jam, . . .

Granularity of Time



Granularity of Position

Raw GPS data

given in degrees (lon/lat) followed by 10 decimals after comma

- example 16.4432840983; 48.2490966797

! too accurate to be used within most comparison-based prediction strategies

Granularity of Position

Raw GPS data

given in degrees (lon/lat) followed by 10 decimals after comma

- example 16.4432840983; 48.2490966797
- ! too accurate to be used within most comparison-based prediction strategies

Truncation

configure accuracy by *truncating* decimals after the comma

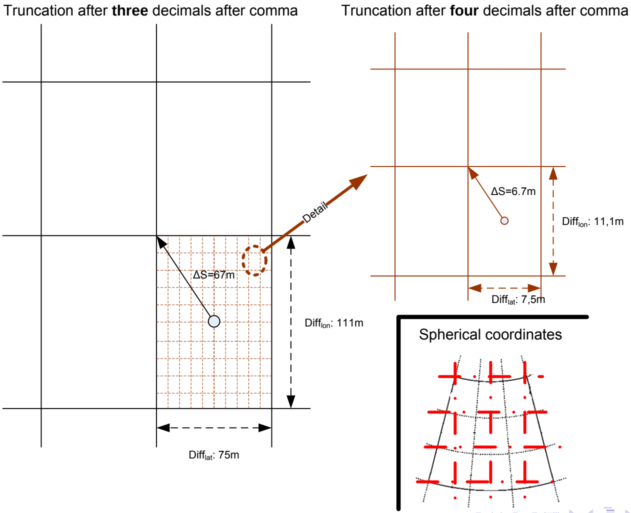
- example 16.443; 48.249

⇒ adapt to positioning systems with different accuracy (GPS, differential GPS, . . .)

⇒ adapt to accuracy needed by the application (pedestrians, vehicle, etc.)

⇒ reduce storage requirements

Granularity of Position



Prediction

Basic comparison strategy

- compare actual and last *i* locations of a client under investigation with history traces
- adapt truncation level if search is not successful

Result

- relative frequencies of next *f* location sequences

⇒ chose location with highest relative frequency

recent sequence	history traces
	...
16,375; 48,178	16,371; 48,183
16,379; 48,177	16,373; 48,181
16,381; 48,177	16,373; 48,181
16,386; 48,176	16,375; 48,179
?	16,375; 48,178
?	16,379; 48,177
	16,381; 48,177
	16,386; 48,176
	16,384; 48,177
	16,383; 48,178
	...

Prediction

Input: pre-processed history set; recent sequence
$S_X = \{X_{c-i}, X_{c-i+1}, \ldots, X_c\}$
Output: prediction of next f location(s)

$\mathcal{P}$:=NULL
repeat
  foreach $S_H = \{H_{c-i}, H_{c-i+1}, .., H_c, .., H_{c+f}\}$ **do**
    if S_X *matches* $S_{H(H_{c-i},..,H_c)}$ **then**
      add S_H to $\mathcal{P}$
    end
  end
  if $\mathcal{P}$==*NULL* **then**
    increase truncation level
  end
until $\mathcal{P}$!=*NULL* ;
chose location with highest relative frequency

Prediction - Alternatives and Advantages

Potential alternative:

- linear extrapolation (based on estimations for the current speed and direction)
- Markov models
- LeZi-Update,. . .

Advantages of our approach:

1. not specific to any type of mobility trace or topology
   - ⇒ can be applied to different mobility patterns (e.g., vehicles, pedestrians, mixed vehicle-pedestrians...)
   - ⇒ the predictor only needs the observed mobility history described in sequences of locations
2. new traces can be easily added without changing predictor
   - ⇒ without rebuilding movement patterns (e.g.,like Markov models)

Outline

Evaluation Objectives

Investigation:

- can replica placement can be improved by adding prediction to the reactive behavior of the FReDi algorithm?

Evaluation Objectives

Investigation:

- can replica placement can be improved by adding prediction to the reactive behavior of the FReDi algorithm?

Several simulations:

- four different placement strategies (*re*- and/or *pro*-active)
- on pre-defined traces
  - fixed movement history traces
  - various fixed evaluation traces

Evaluation Objectives

Investigation:

- can replica placement can be improved by adding prediction to the reactive behavior of the FReDi algorithm?

Several simulations:

- four different placement strategies (*re*- and/or *pro*-active)
- on pre-defined traces
  - fixed movement history traces
  - various fixed evaluation traces

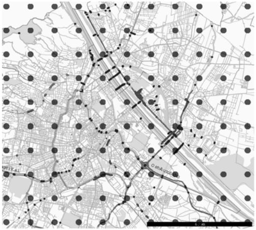
Measurements:

- first comparison of introduced predictor (PR) with other prediction models
- accuracy and improvement of replica placement algorithm

Simulation Setup

Access point infrastructure modeled as a square grid ($N \times N$)

- mapped onto the city of Vienna
- assumption: one access point per proxy cache
- taxis (clients) access replicas held by PCs
- taxis are equipped with mobile devices and GPS receivers
- each taxi sends requests to its geographically nearest PC
- based on real-world GPS traces of taxi moving in the city of Vienna



City map of Vienna, Austria

Example: Vienna

Simulation Setup

About 18 500 different location traces (after preprocessing)

- average length of 105 locations per trace

⇒ nearly 2 million different GPS locations overall

- for evaluation: 400 traces (each with a length of 200 positions)

⇒ resulting in 80 000 locations for evaluation

Granularity of time (gt):

- Δgt = 20 sec.

Granularity of position (gp):

- truncation set to three decimals after comma

⇒ for Vienna (lat.: 75m accuracy, lon.: 111m accuracy)

Sequence length:

- use actual and last 3 locations for comparison

Investigation

Evaluation process:

Manage the placement of one replica for each taxi independently

this allows to:

- evaluate whether the accuracy of placement can be improved by using prediction
- determine which taxi characteristics might influence this improvement

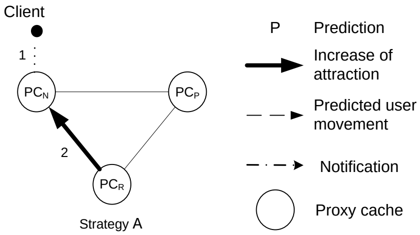
⇒ four different placement strategies

Placement Strategies

Strategy A - Actual

- depends only on current taxi position
- represents standard behavior of FReDi

⇒ ***reactive*** replica placement



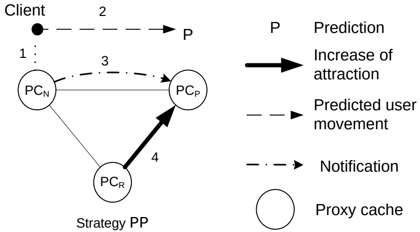
`Step 1:` client determines its nearest proxy cache PC_N
`Step 2:` update the attraction in the direction of PC_N

Placement Strategies

Strategy PP - Perfect Prediction

- not really a prediction
- perfect knowledge about the next position

⇒ reference strategy

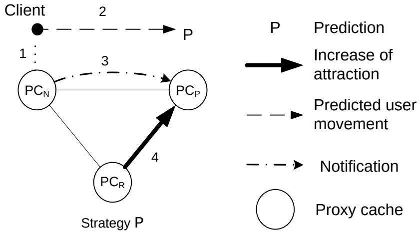


`Step 1:` client determines its nearest proxy cache PC_N
`Step 2:` determine the clients next location
`Step 3:` notify the predicted next nearest proxy cache PC_P
`Step 4:` update the attraction in the direction of PC_P

Placement Strategies

Strategy P - Prediction

- uses **only prediction** to place replica
- resulting activities are similar as described for strategy PP

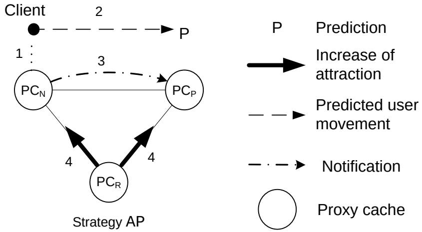


Step 1: client determines its nearest proxy cache PC_N
Step 2: determine the clients next location
Step 3: notify the predicted next nearest proxy cache PC_P
Step 4: update the attraction in the direction of PC_P

Placement Strategies

Strategy AP - Actual and Prediction

- uses current position **and** prediction
- combines the Strategies A and P



Step 1: client determines its nearest proxy cache PC_N

Step 2: determine the clients next location

Step 3: notify the predicted next nearest proxy cache PC_P

Step 4: update the attraction halfway towards PC_N and PC_P

Outline

1. Replica Placement Algorithm (FReDi)
2. Mobility Prediction
3. Evaluation
4. Results
   - Prediction Accuracy
   - Replica Placement Accuracy
5. Conclusion

Prediction Accuracy

Error of prediction

- in terms of the (geo) distance between *predicted* and *actual* future position of evaluation traces

	Two decimals	Three decimals
PR [1]	172m	143m
LeZi-Update [2]	405m	208m
Markov [3]	411m	223m

Table: Predictor accuracy - average error

[1] `PR`: self developed predictor (PR)

[2] `LeZi-Update`: based on dictionaries of individual user's path updates

[3] `Markov`: forth order Markov predictor (implementing subsequent decrease of order in case no fitting history could be found for current order)

Single Replica Placement Accuracy

The accuracy of a **single** replica placement can be measured by the *distance* between…

- a client (taxi) requesting the data and
- a replica of the data of interest

Single Replica Placement Accuracy

The accuracy of a **single** replica placement can be measured by the *distance* between...

- a client (taxi) requesting the data and
- a replica of the data of interest

More detailed:
Calculated as distance d_{PC} between...

- the taxis nearest PC and
- the nearest PC storing a replica of this data

measured in multiples of PC hops

Overall Replica Placement Accuracy

The overall placement accuracy is assessed by

- the mean distance d of all d_{PC} values
- for all taxis calculated for each time step of the simulation period

$$d = \frac{\sum_{t \in TS} \sum_{id \in T} (d_{PC}(l_{id,t}, r_{id,t}))}{|T||TS|}, \qquad (1)$$

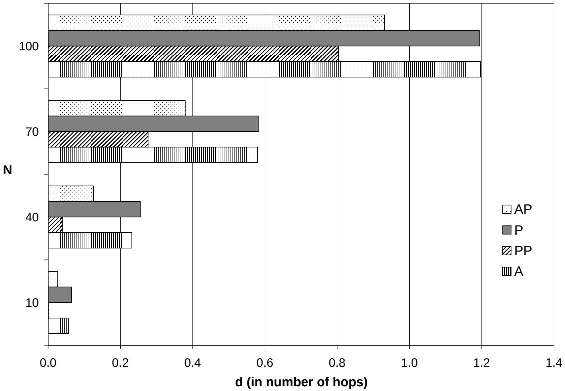
where ...

- TS is the set of time steps
- T is the set of taxis
- $l_{id,t}$ is the location of the nearest PC to the taxi *id* at time step t
- $r_{id,t}$ is the location of the nearest PC holding the replica of interest for taxi *id* at time step t

Overall Replica Placement Accuracy

Overall distance d of the different strategies for varying N

N is the size of square ($N \times N$) grid



Conclusion

Investigation:

can performance of replica placement algorithms be improved by adding *proactive* strategies based on predictions of client movements?

Answer: Yes, ... *but*

strongly influenced by quality and accuracy of prediction

- strategy based on *perfect prediction* (PP) – up to 100% improvement
- strategy based on prediction only (P) – no real improvements
- combined strategy (actual *and* predicted position) (AP) – about 80% improvement

(over reactive strategy A)

Ongoing and Future Work

Improving and evaluating the prediction component

in terms of accuracy

in terms of computational cost

Various traces from different applications

varying accuracy (pedestrians, vehicular, . . .)

Client groups (communities) interested in the same content

instead of individual clients

Apply proactive movement strategies in combination with *simpler* replica placement algorithms

Acknowledgements

This work was supported by:

The AMADEUS Project

Founded by OEAD – Austrian Exchange Service