SECENTIS

A European Industrial Doctorate on Security and Trust

VLOC: Verifying the Physical Location of A Virtual Machine In Cloud

Mojtaba Eskadnari

Advisors:

- Bruno Crispo,

- Anderson Santana de Oliveira

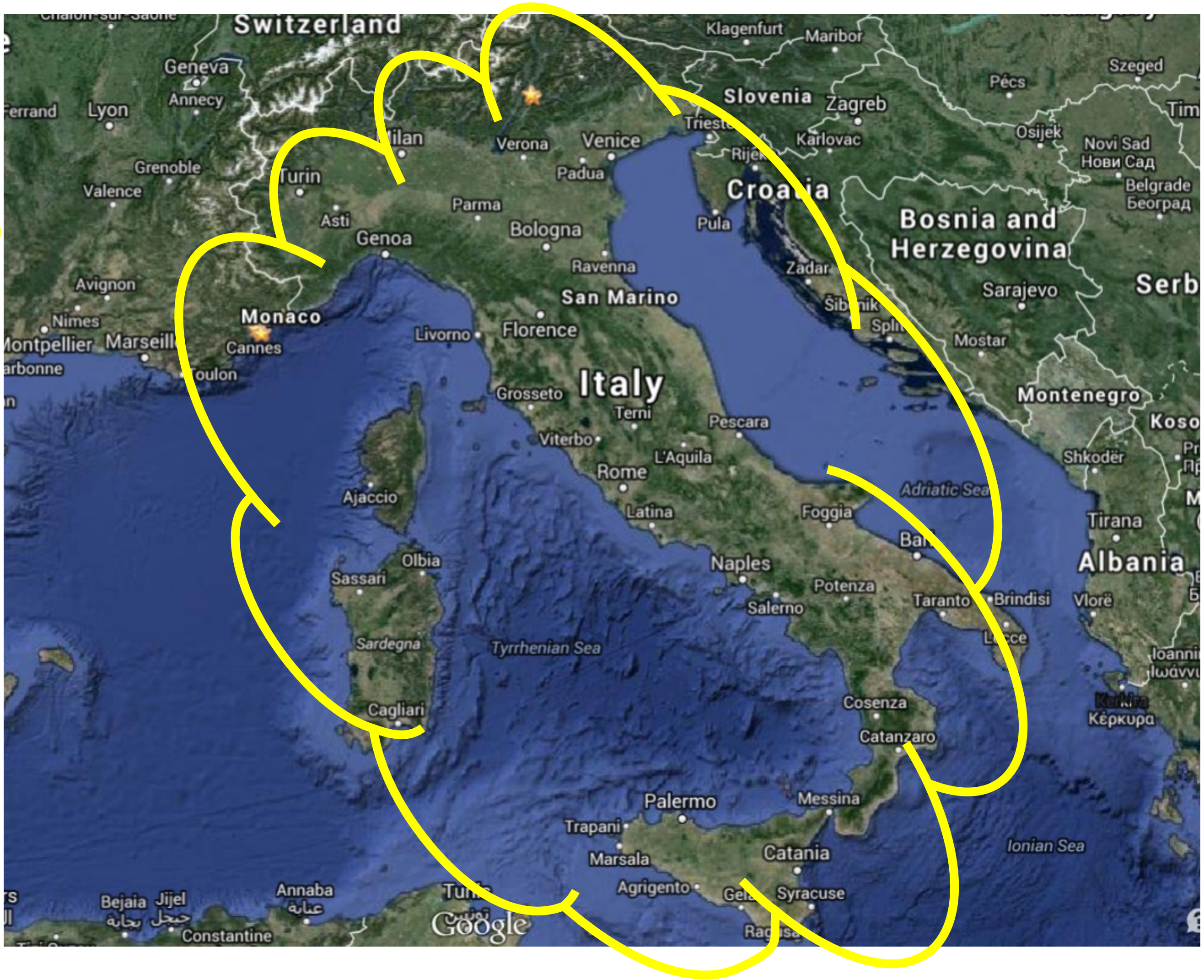
[Oct 2014]

FONDAZIONE BRUNO KESSLER

UNIVERSITY OF TRENTO - Italy

Your data is here

Maybe it should be here



Motivation

- Is data within some **political** boundary
- **Privacy** protections
- **Intellectual property** protections
- Regulatory **compliance**

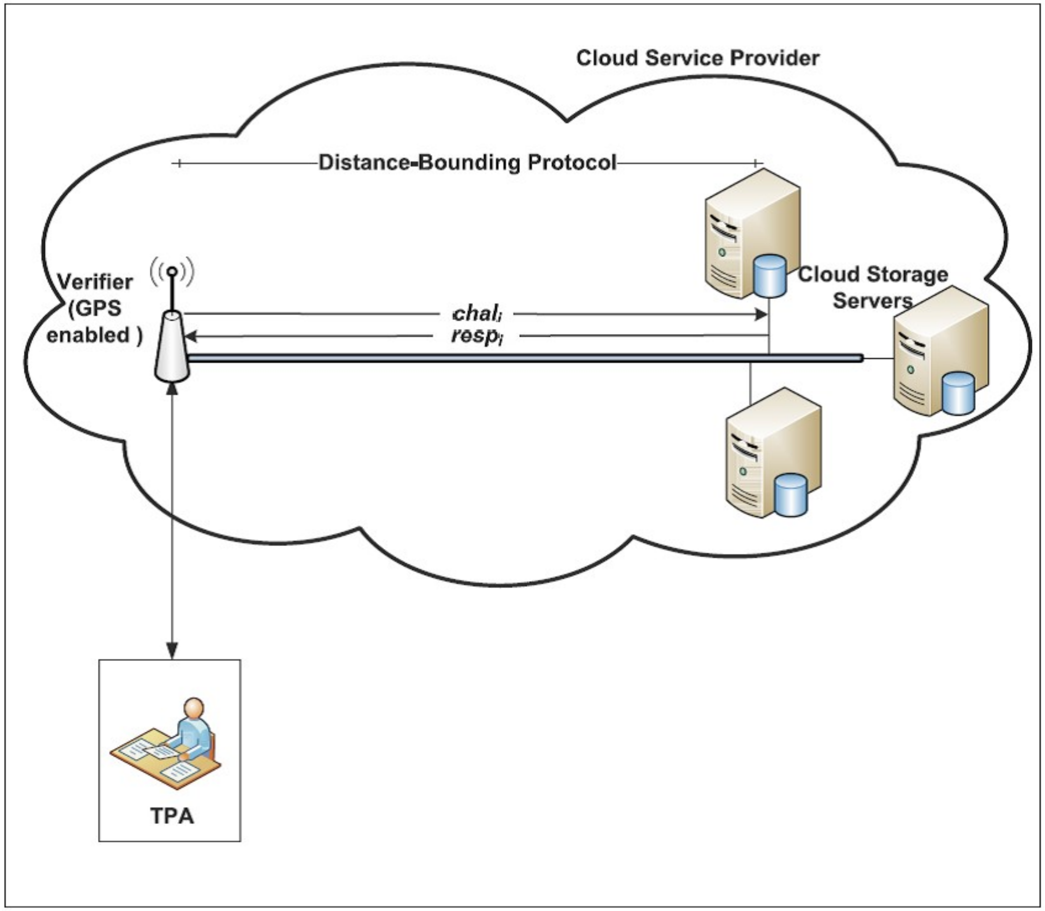
What is the problem?

What is the problem?

What is the problem?

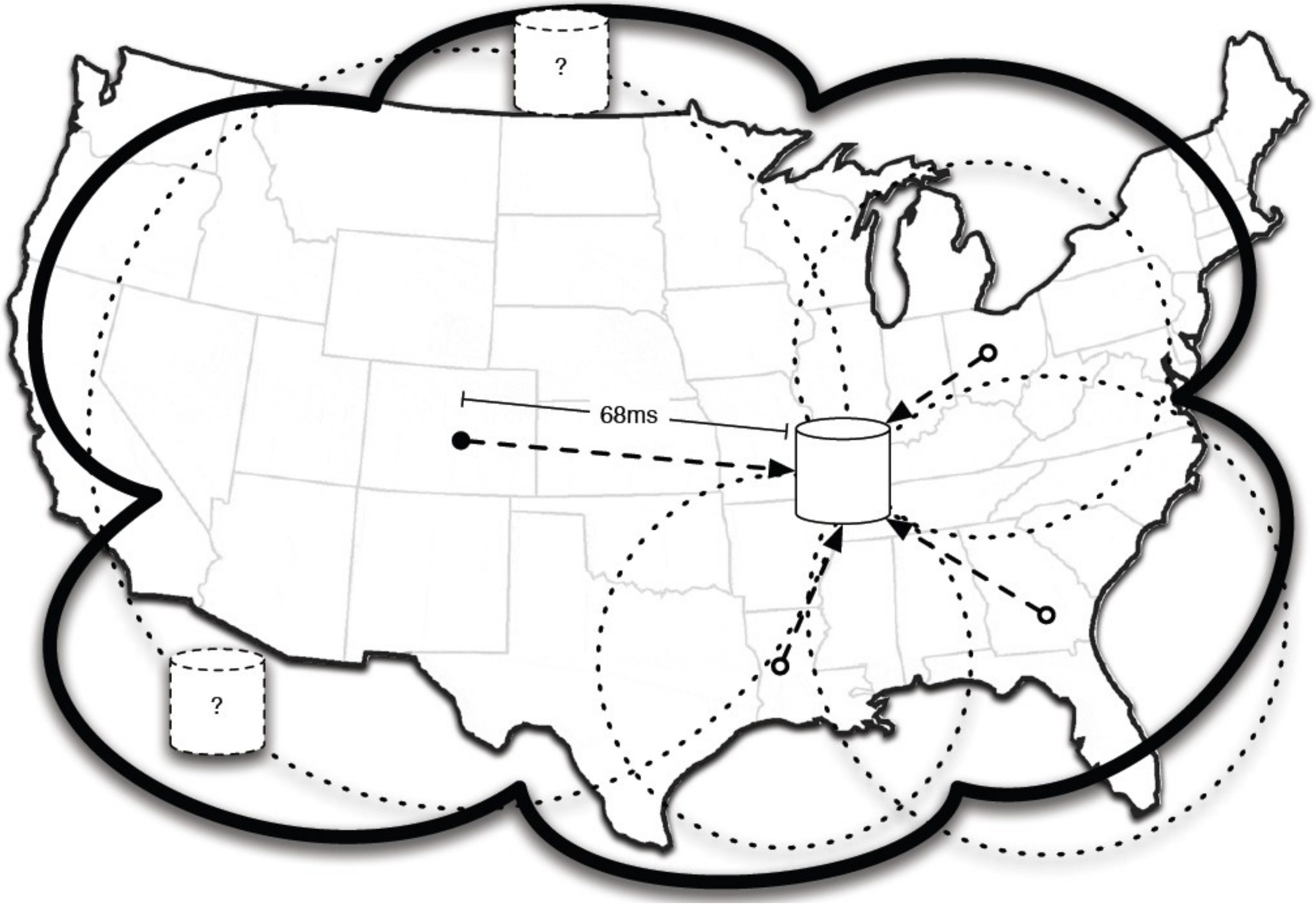
State of the Art

Using a GPS enabled device



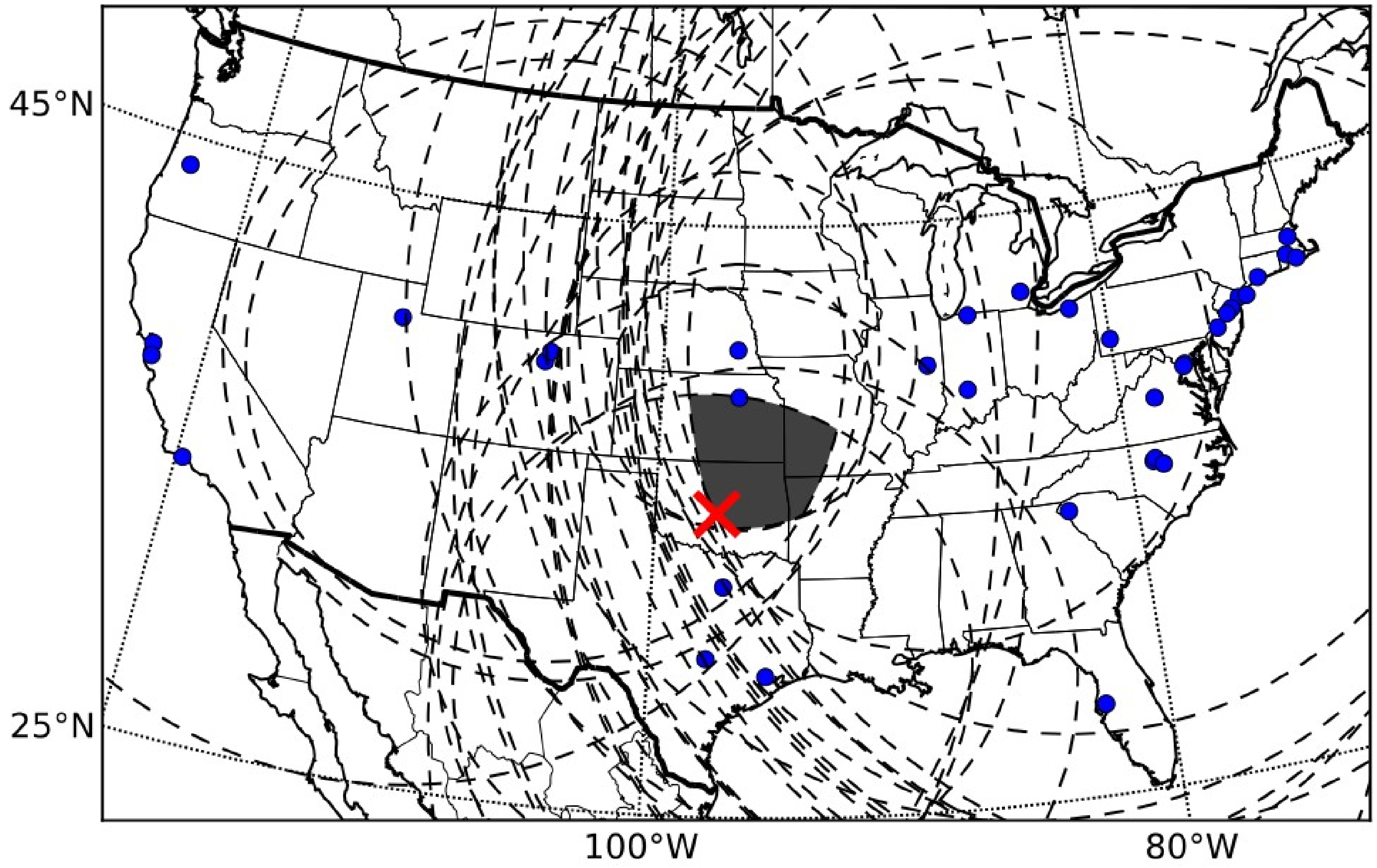
A. Albeshri, C. Boyd, and J. Nieto, “**GeoProof**: Proofs of geographic location for cloud computing environment,” *in Distributed Computing Systems Workshops (ICDCSW)*, 2012

Latency-based measurement



Peterson et al., “The importance of geo-locating data in the cloud” in *USENIX on HotCloud 2011*

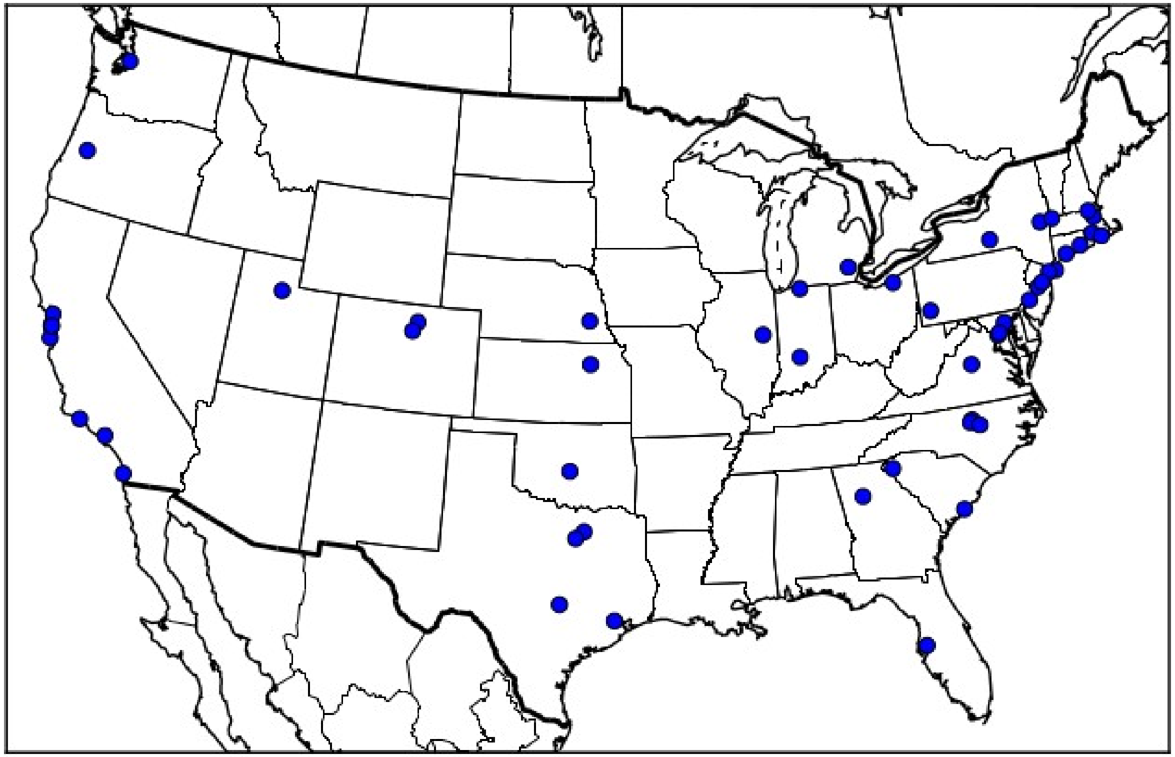
Latency-based with Observation



M. Gondree and Z. N. Peterson, "GeoLocation of data in the cloud," *in CODASPY* 2013.

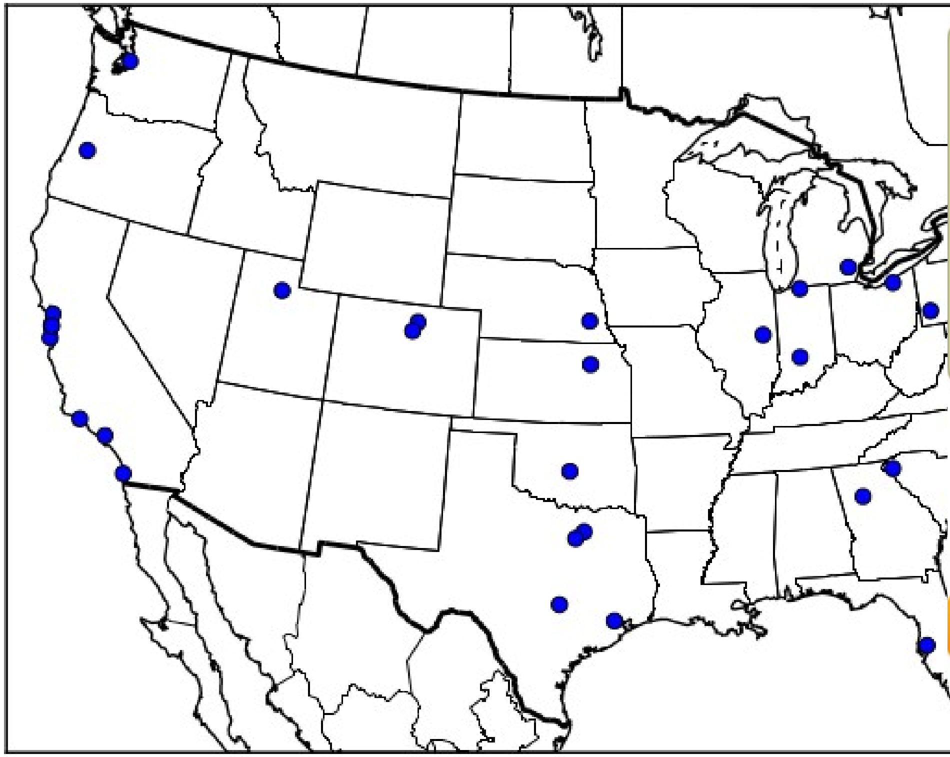
The Main Drawbacks

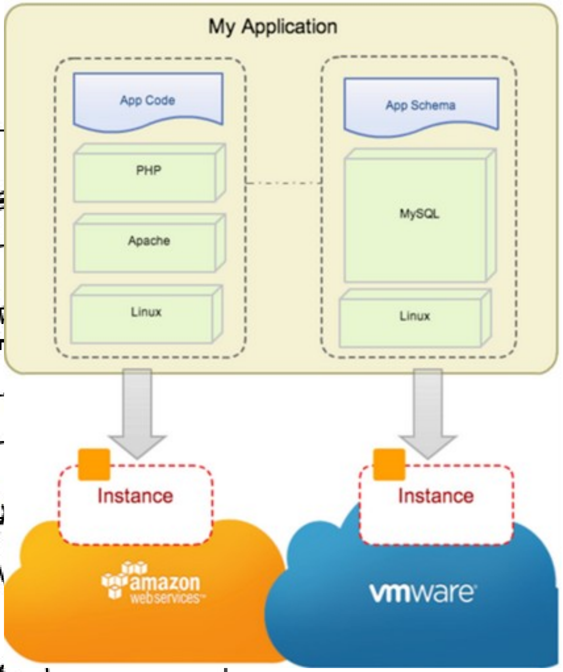
- Various service layers (App, Platform, OS, …)
- Requires hardware equipments (*i.e.* external landmarks)



The Main Drawbacks

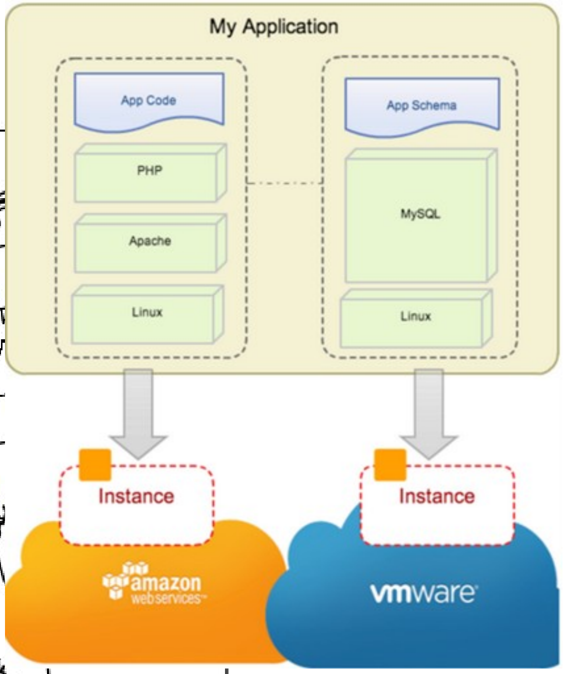
- Various service layers (App, Platform, OS, …)
- Requires hardware equipments (*i.e.* external landmarks)





The Main Drawbacks

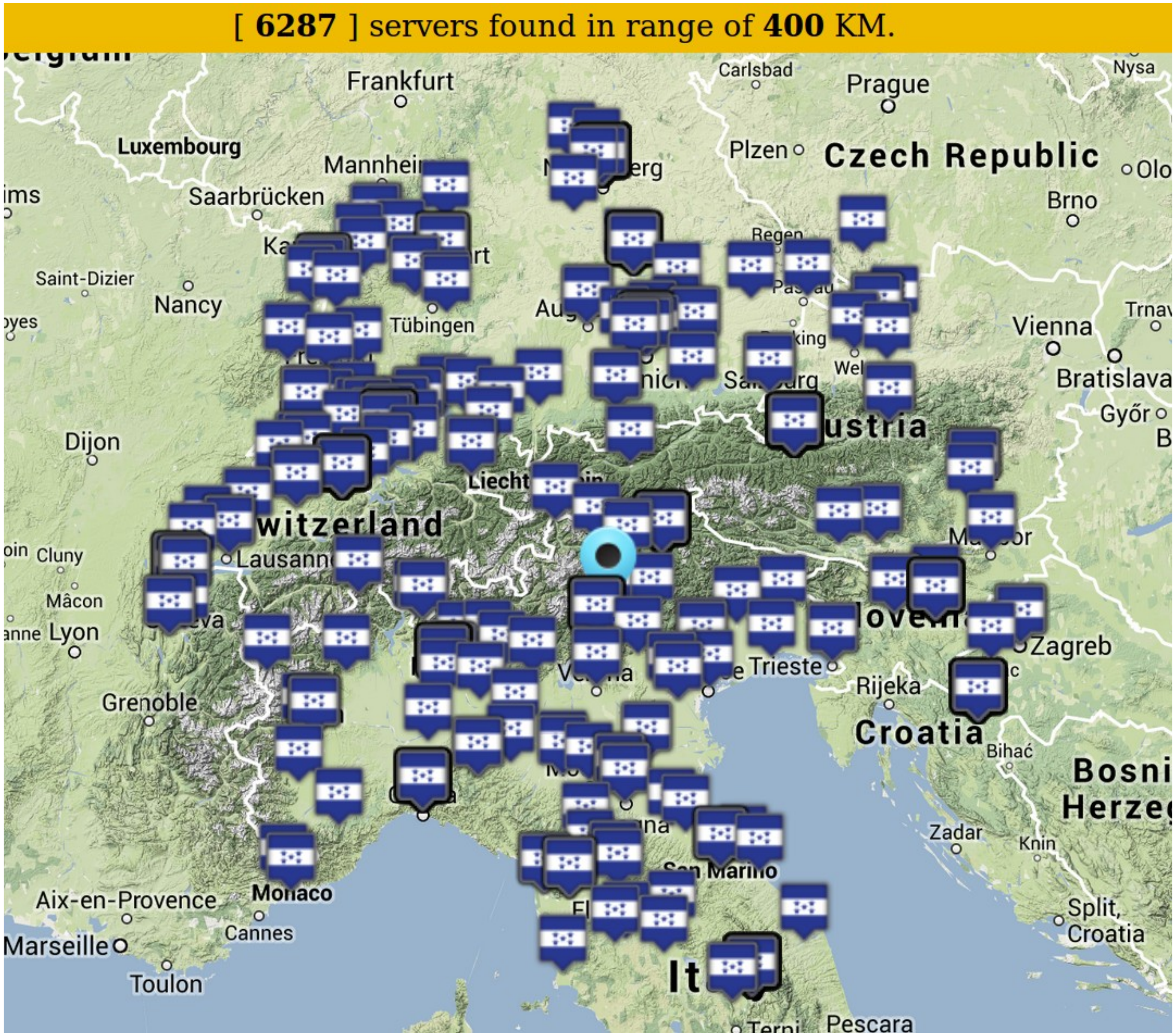
- Various service layers (App, Platform, OS, …)
- Requires hardware equipments (*i.e.* external landmarks)



What is VLOC?

- A tool installed on VM
- Initialised automatically
- Adapts to the environment
- Verifies its physical location

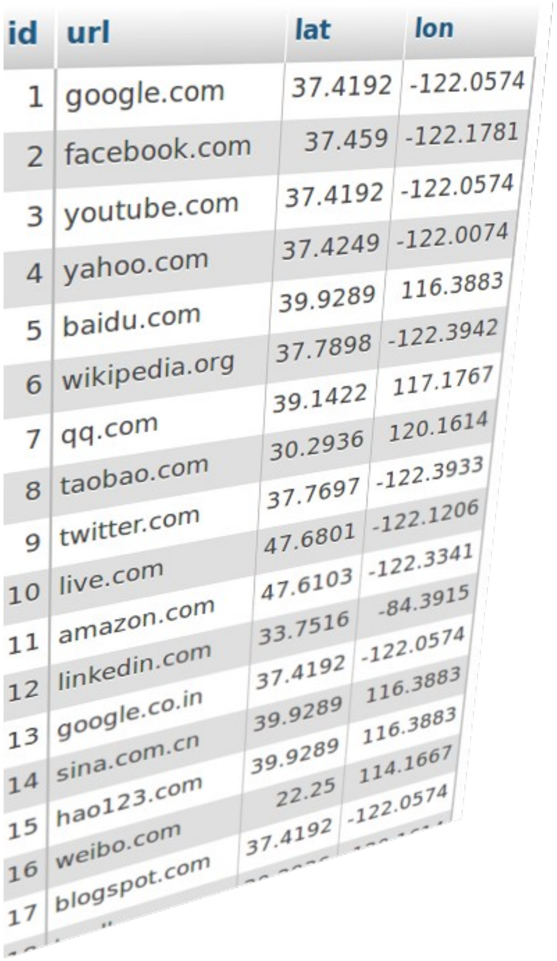
How it works (1): *Finding servers*



[**6287**] servers found in range of **400** KM.

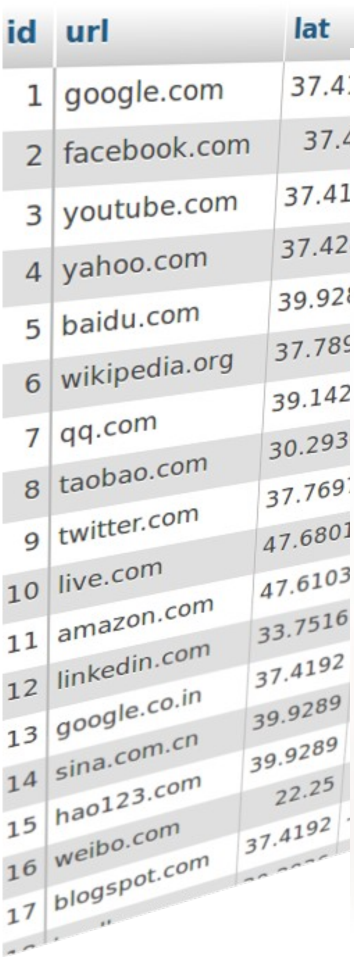
How it works (2): *Coordinates*

IPaddressAPI.com



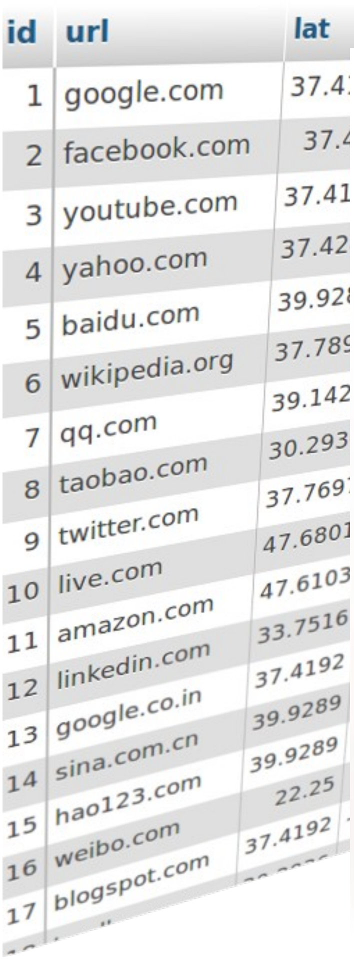
id	url	lat	lon
1	google.com	37.4192	-122.0574
2	facebook.com	37.459	-122.1781
3	youtube.com	37.4192	-122.0574
4	yahoo.com	37.4249	-122.0074
5	baidu.com	39.9289	116.3883
6	wikipedia.org	37.7898	-122.3942
7	qq.com	39.1422	117.1767
8	taobao.com	30.2936	120.1614
9	twitter.com	37.7697	-122.3933
10	live.com	47.6801	-122.1206
11	amazon.com	47.6103	-122.3341
12	linkedin.com	33.7516	-84.3915
13	google.co.in	37.4192	-122.0574
14	sina.com.cn	39.9289	116.3883
15	hao123.com	39.9289	116.3883
16	weibo.com	22.25	114.1667
17	blogspot.com	37.4192	-122.0574

How it works (3): *Latency measurement*

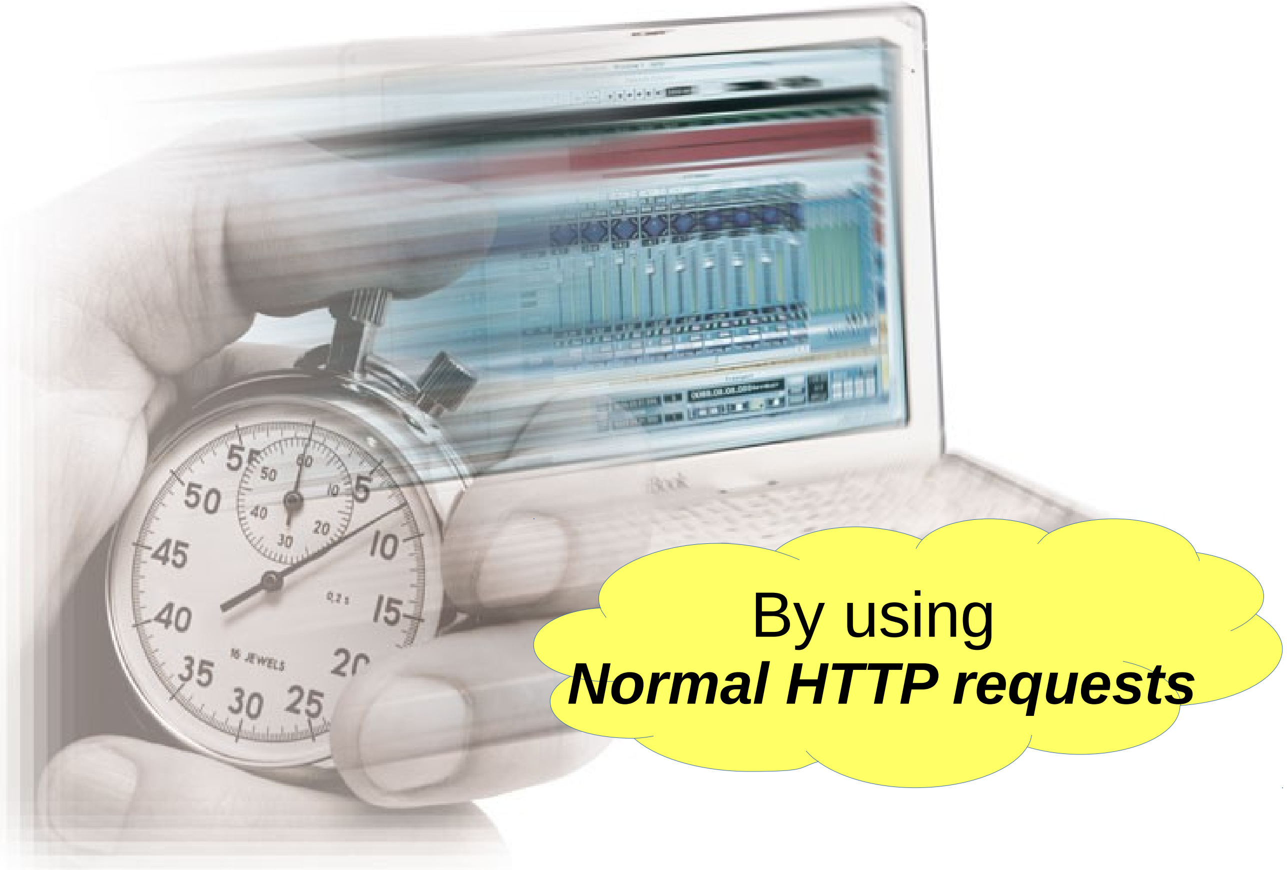


id	url	lat	lon
1	google.com	37.4	
2	facebook.com	37.4	
3	youtube.com	37.41	
4	yahoo.com	37.42	
5	baidu.com	39.928	
6	wikipedia.org	37.789	
7	qq.com	39.142	
8	taobao.com	30.293	
9	twitter.com	37.769	
10	live.com	47.6801	
11	amazon.com	47.6103	
12	linkedin.com	33.7516	
13	google.co.in	37.4192	
14	sina.com.cn	39.9289	
15	hao123.com	39.9289	
16	weibo.com	22.25	
17	blogspot.com	37.4192	

How it works (3): *Latency measurement*

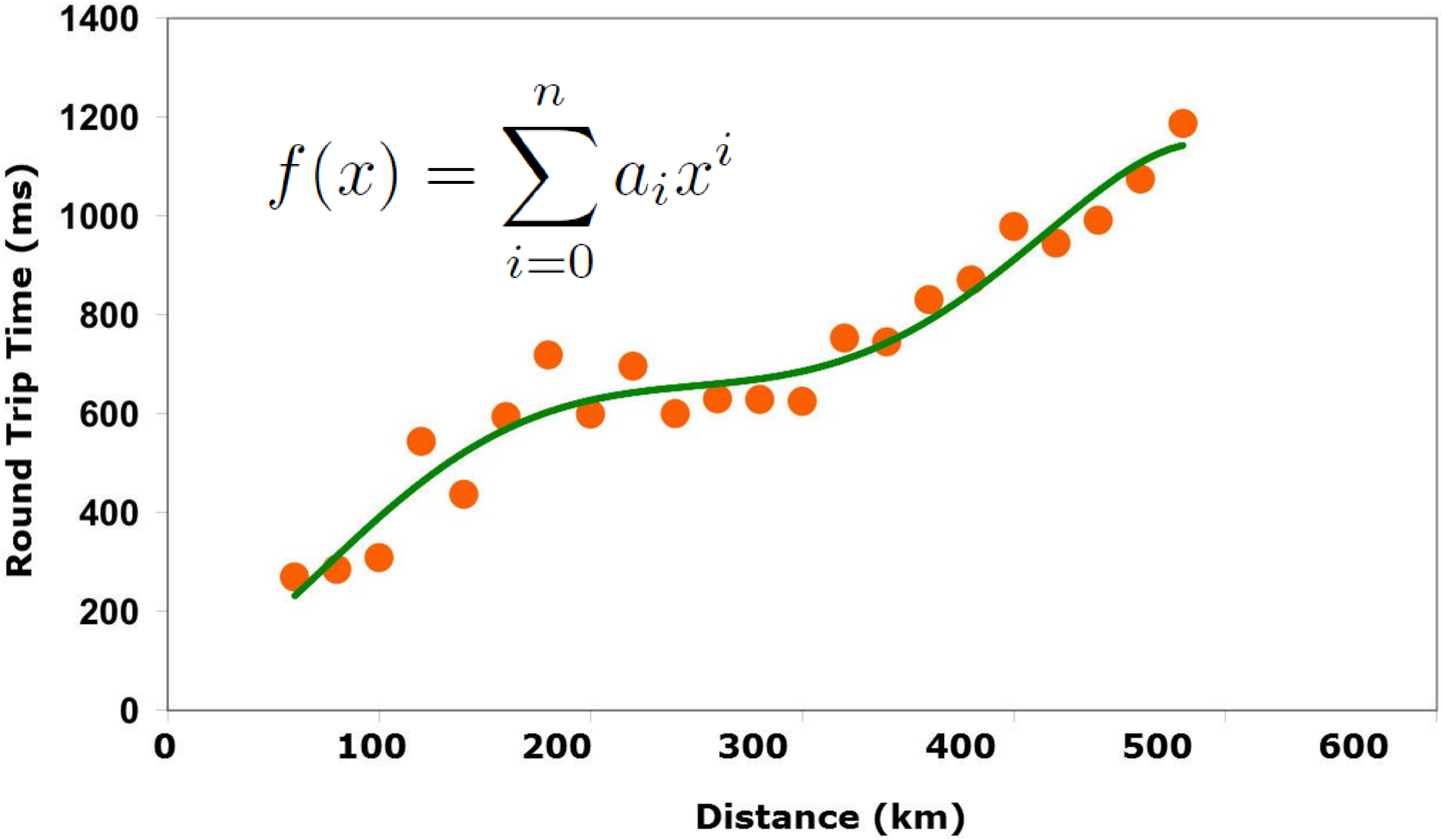


id	url	lat	lon
1	google.com	37.4	
2	facebook.com	37.4	
3	youtube.com	37.41	
4	yahoo.com	37.42	
5	baidu.com	39.92	
6	wikipedia.org	37.78	
7	qq.com	39.142	
8	taobao.com	30.293	
9	twitter.com	37.769	
10	live.com	47.6801	
11	amazon.com	47.6103	
12	linkedin.com	33.7516	
13	google.co.in	37.4192	
14	sina.com.cn	39.9289	
15	hao123.com	39.9289	
16	weibo.com	22.25	
17	blogspot.com	37.4192	



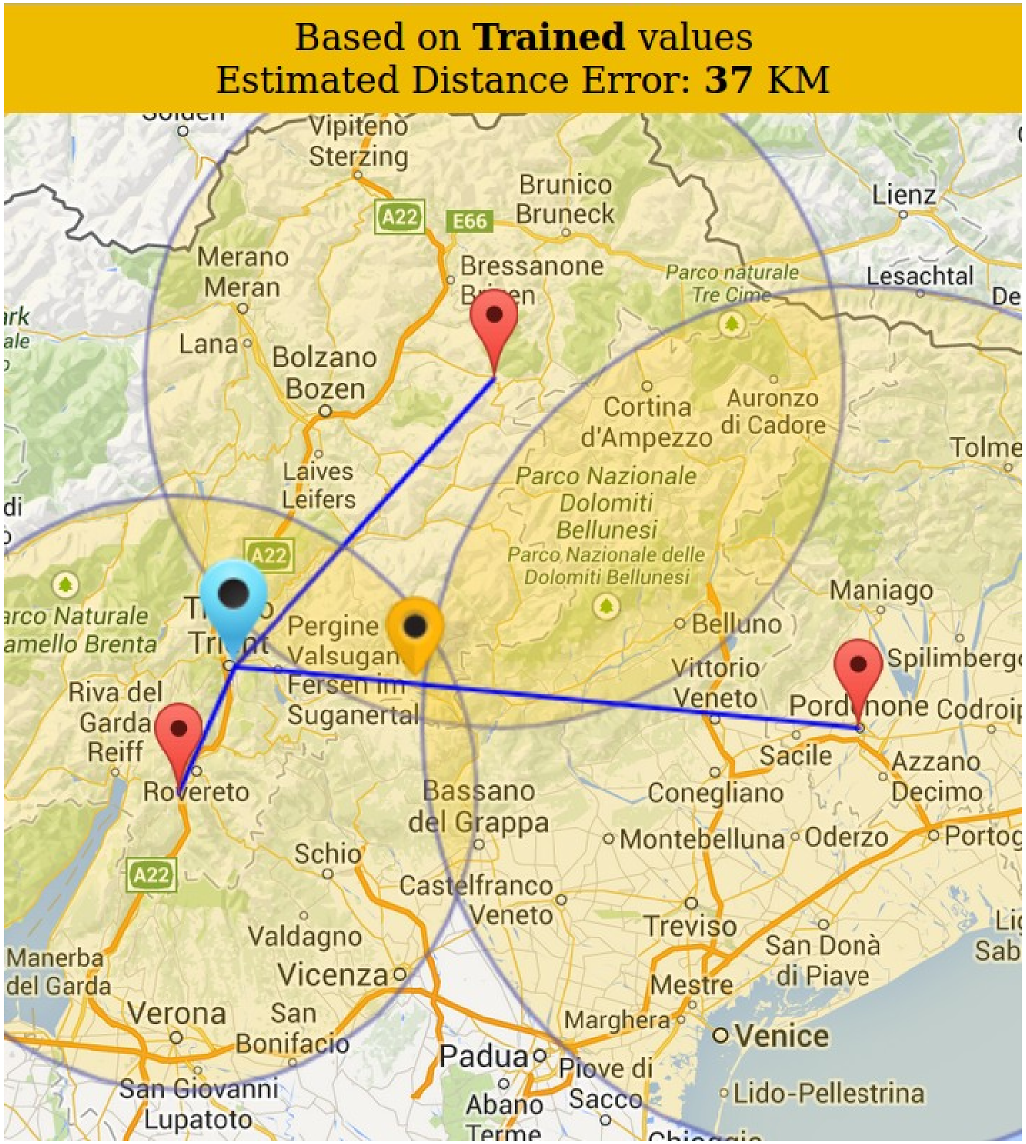
By using
Normal HTTP requests

How it works (4): *Training*



$$f(x) = \sum_{i=0}^{n} a_i x^i$$

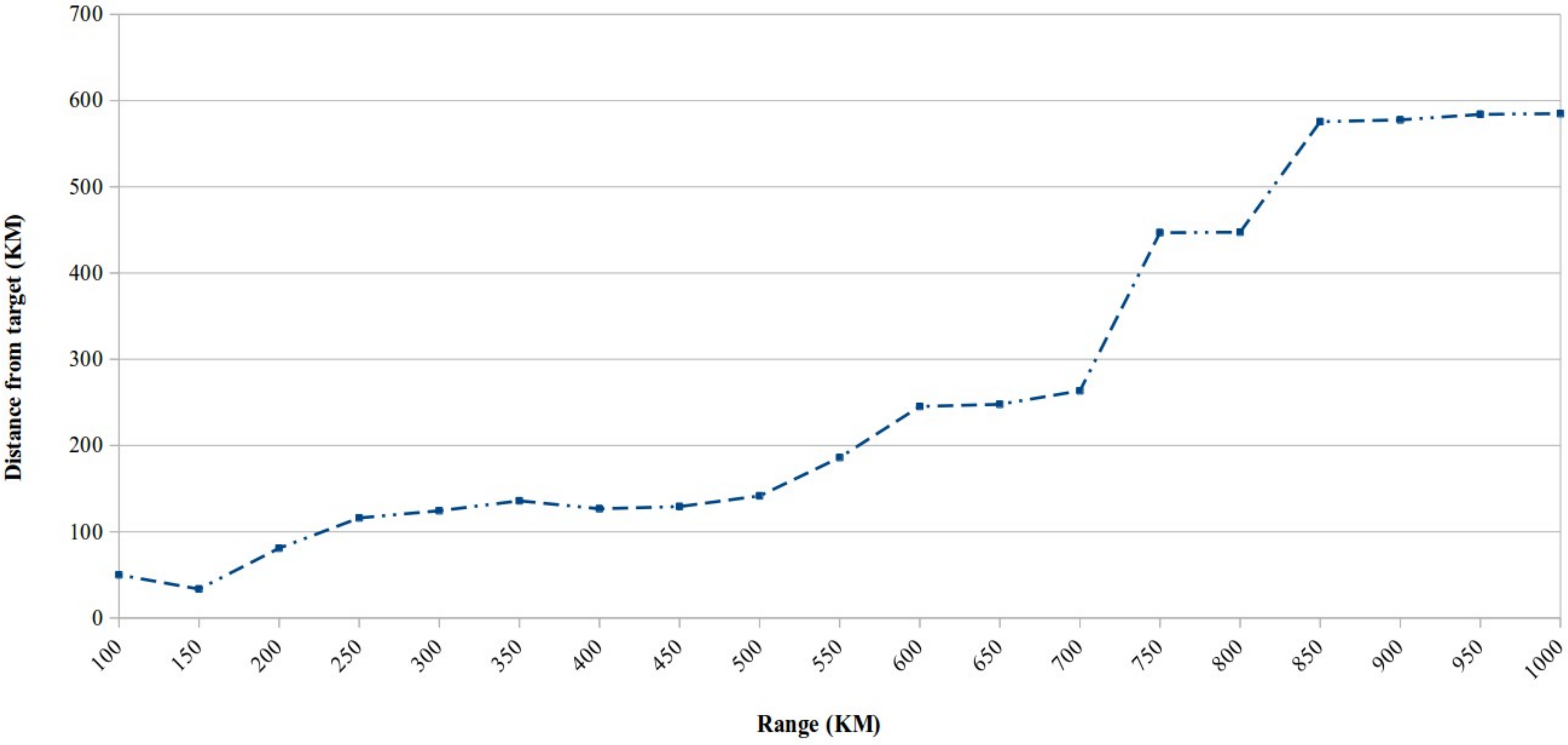
How it works (5): *Localisation*



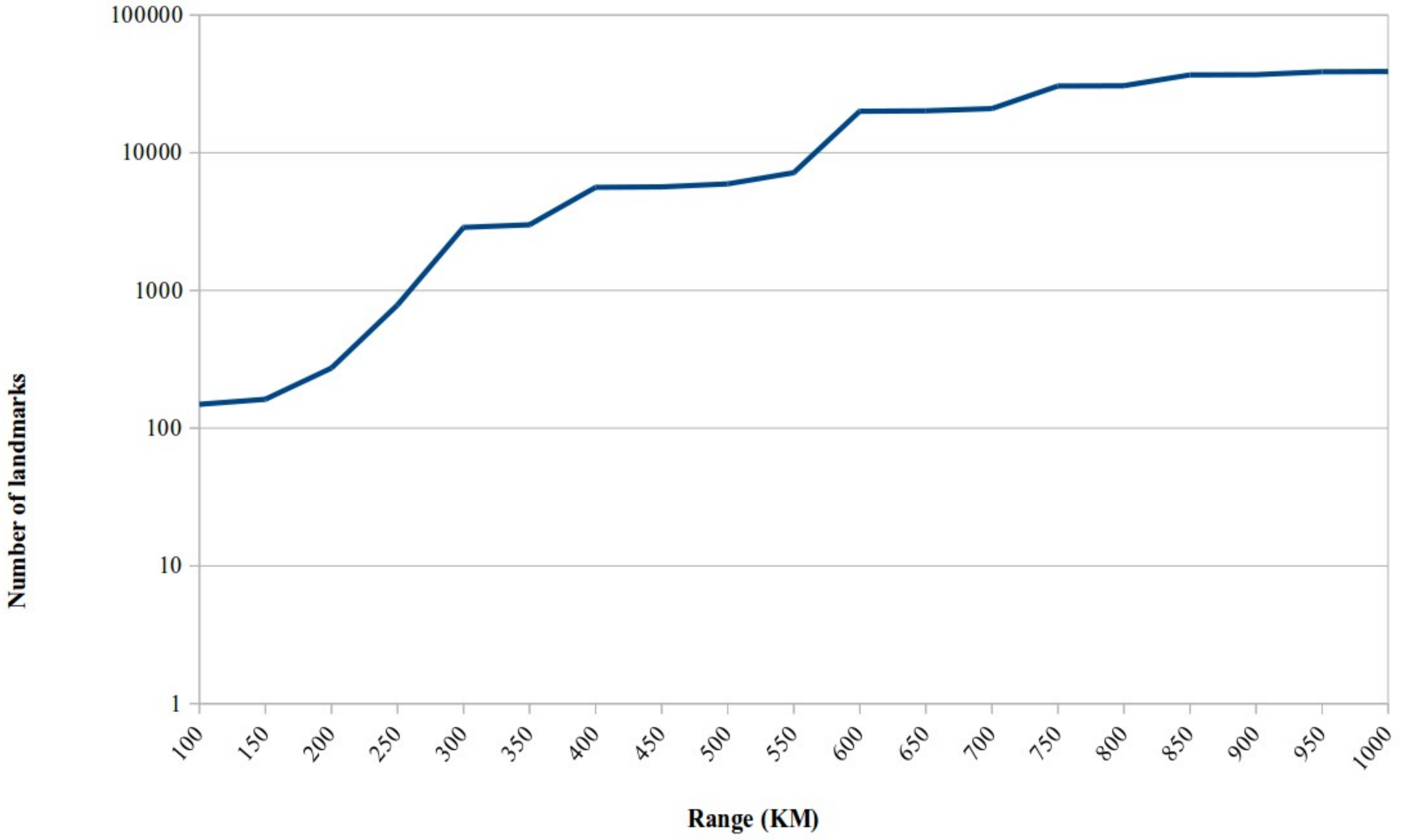
Based on **Trained** values

Estimated Distance Error: **37** KM

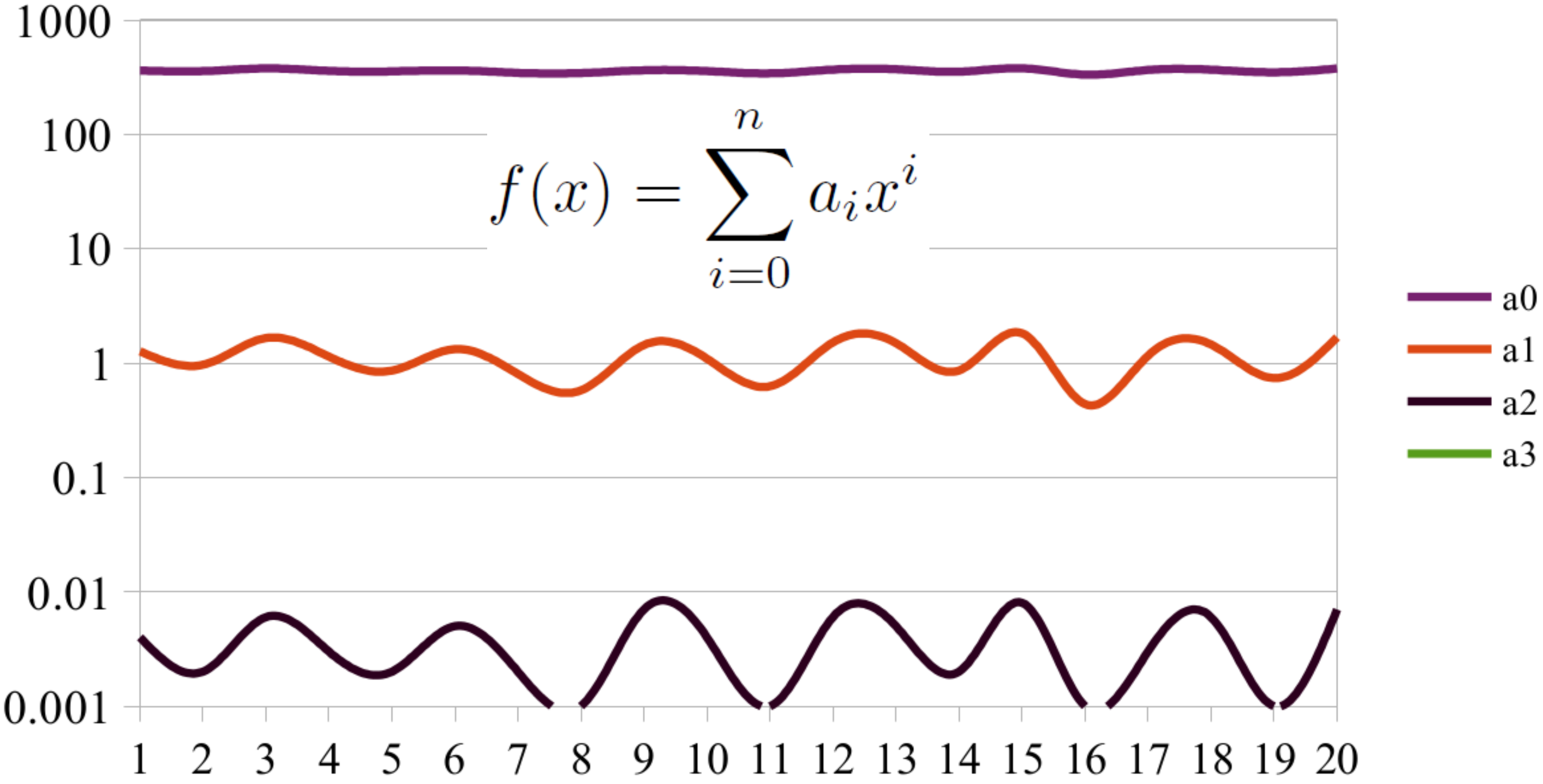
Experimental Results (1): *Accuracy*



Experimental Results (2): *No. of Landmarks*



Experimental Results (3): *Coefficients*



$$f(x) = \sum_{i=0}^{n} a_i x^i$$

Update process captured ***20*** times

Limitations

- Data must **not** be moved during **initialisation**
- Network **latency changes**

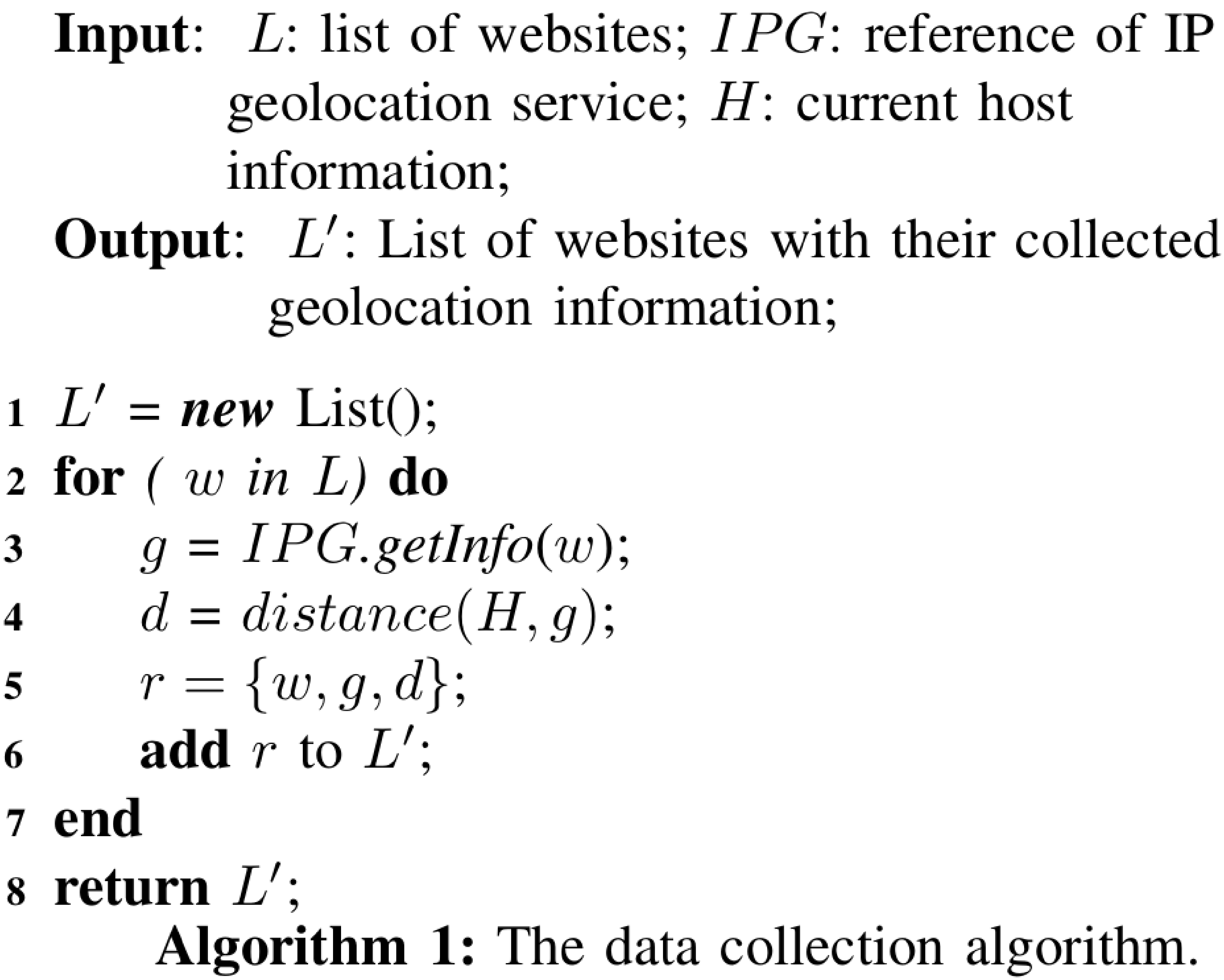
  (*frequent observation is required*)
- **Parameter tunning** needed to achieve the best accuracy
- For **very close regions**, it is not accurate enough.

  (*e.g. north of France and Belgium*)

Conclusions

- Does not require any particular hardware
- Does not require a fixed network of landmarks
- Adaptable to dynamic environments like the Internet
- Deployed as a software tool
- Can be employed in monitoring and enforcement of location based policies.
- Does not relay to the trustworthiness of the cloud provider

Email: eskandari@fbk.eu

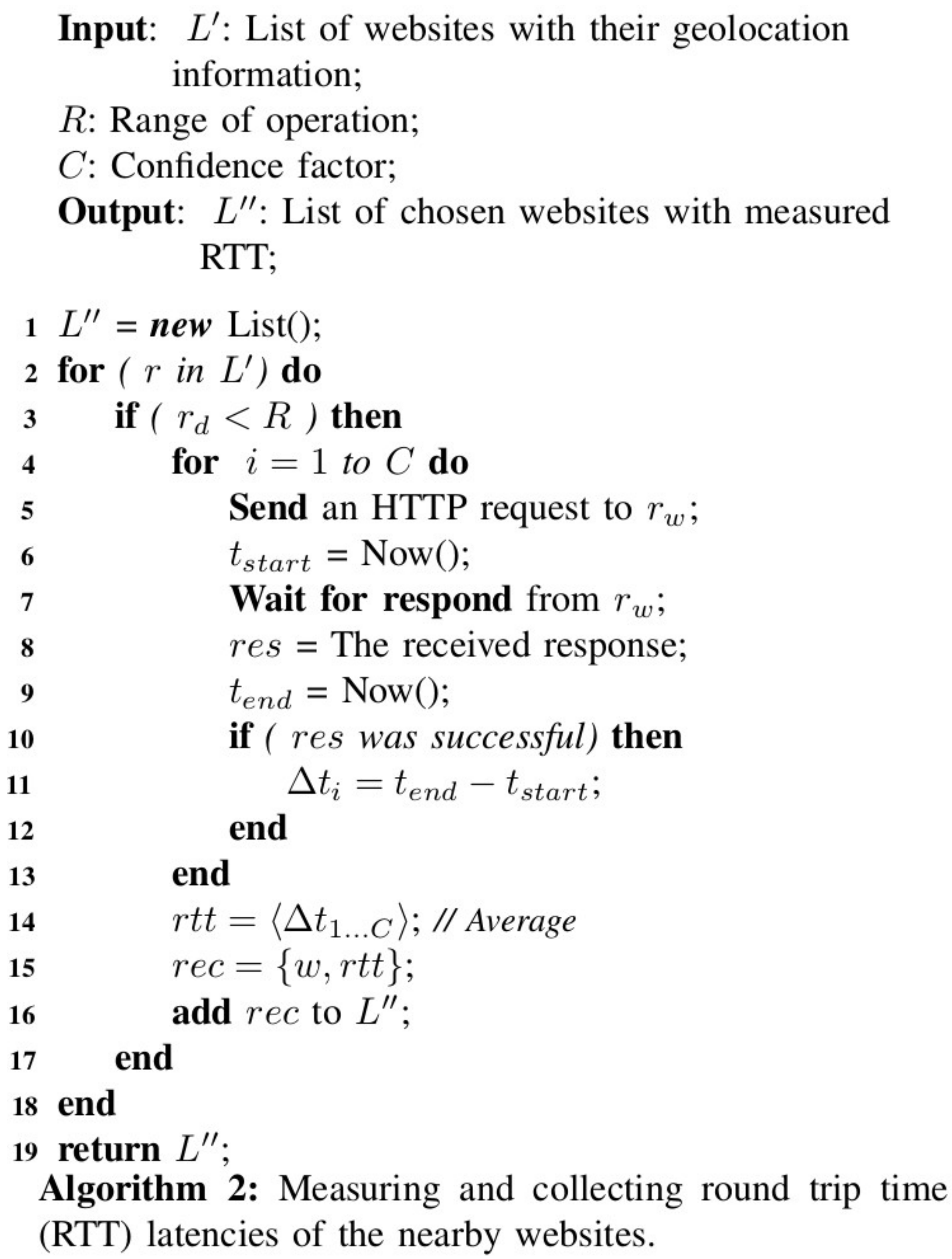


Input: L: list of websites; IPG: reference of IP geolocation service; H: current host information;

Output: L': List of websites with their collected geolocation information;

```
$L'$ = *new* List();
**for** ( $w$ *in* $L$) **do**
    $g = IPG.getInfo(w)$;
    $d = distance(H, g)$;
    $r = \{w, g, d\}$;
    **add** $r$ to $L'$;
**end**
**return** $L'$;
```

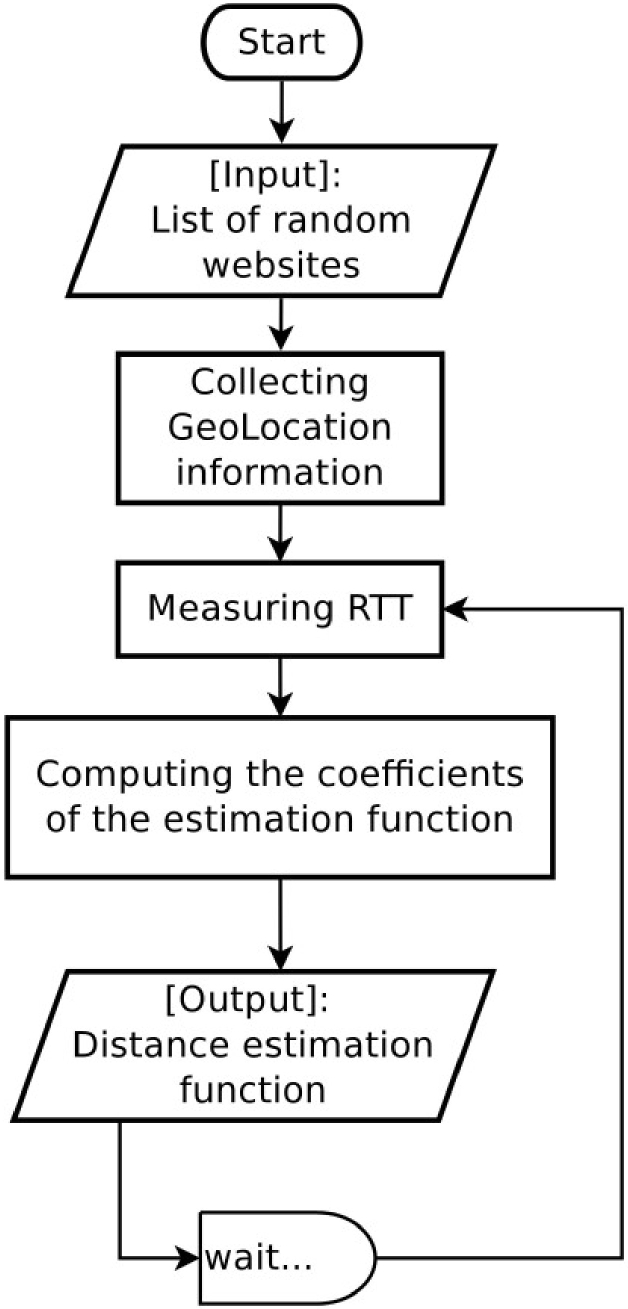
Algorithm 1: The data collection algorithm.



Input: L': List of websites with their geolocation information;
R: Range of operation;
C: Confidence factor;
Output: L'': List of chosen websites with measured RTT;

```
$L''$ = new List();
for ( $r$ in $L'$ ) do
    if ( $r_d < R$ ) then
        for $i = 1$ to $C$ do
            Send an HTTP request to $r_w$;
            $t_{start}$ = Now();
            Wait for respond from $r_w$;
            $res$ = The received response;
            $t_{end}$ = Now();
            if ( $res$ was successful) then
                $\Delta t_i = t_{end} - t_{start}$;
            end
        end
        $rtt = \langle \Delta t_{1 \ldots C} \rangle$; // Average
        $rec = \{w, rtt\}$;
        add $rec$ to $L''$;
    end
end
return $L''$;
```

Algorithm 2: Measuring and collecting round trip time (RTT) latencies of the nearby websites.



Start

[Input]:
List of random websites

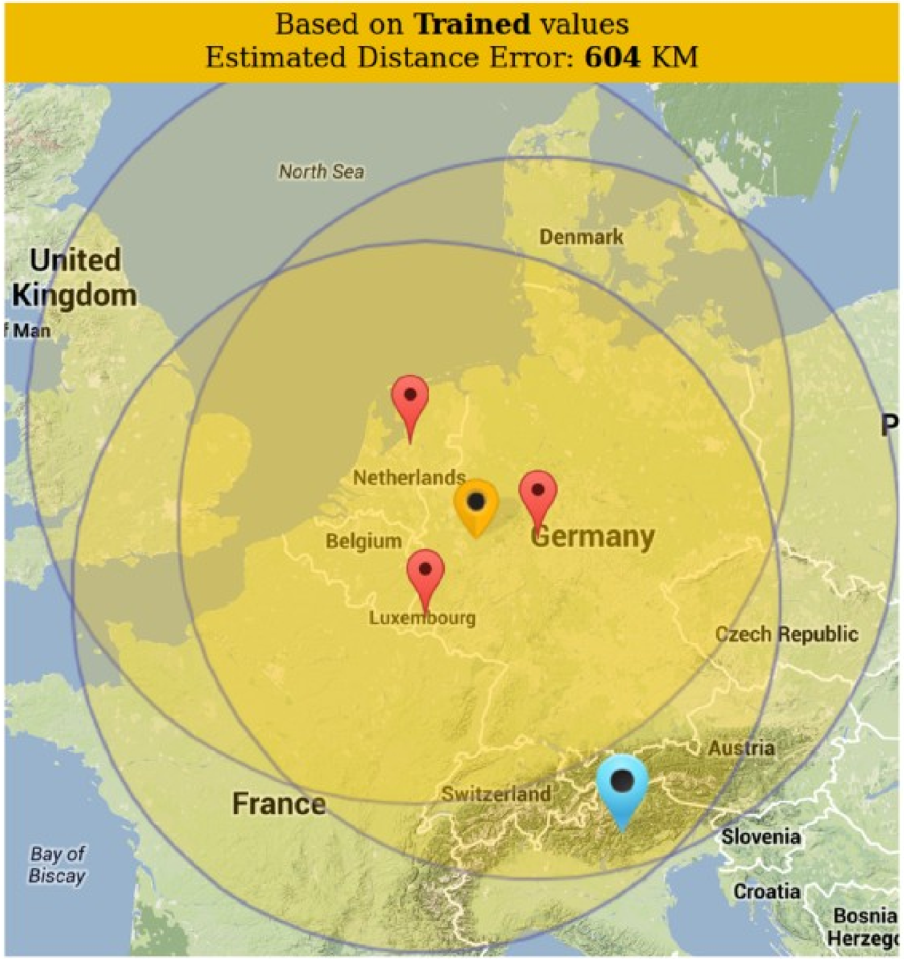
Collecting GeoLocation information

Measuring RTT

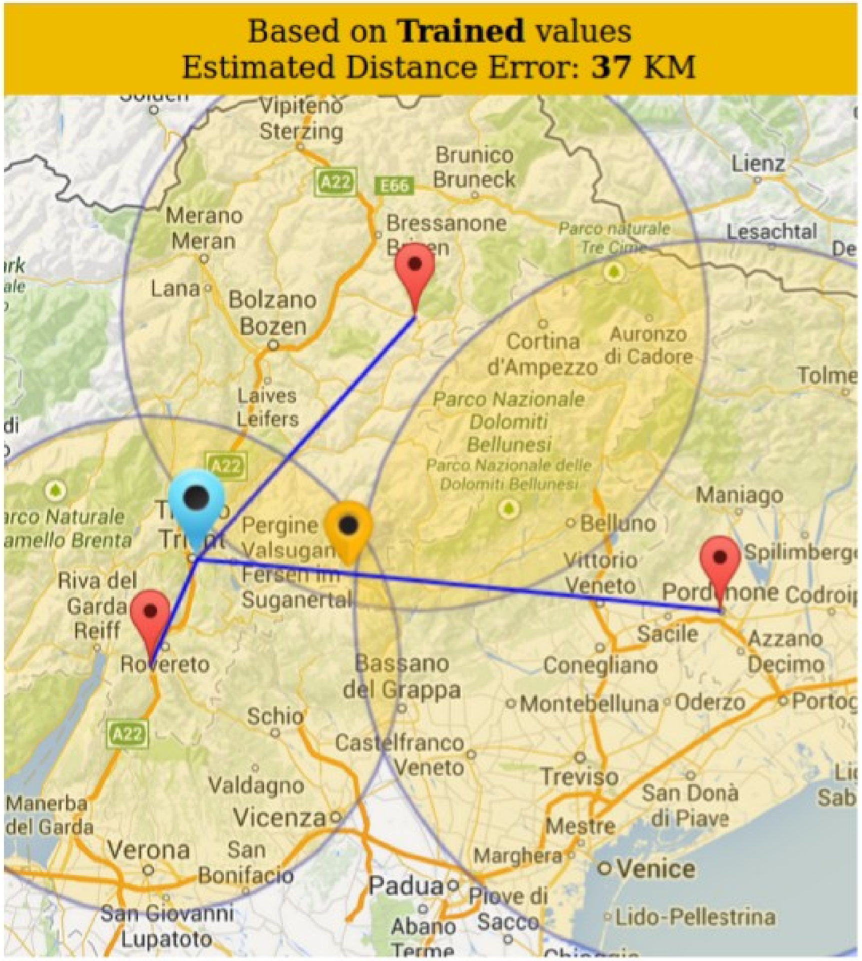
Computing the coefficients of the estimation function

[Output]:
Distance estimation function

wait...



(a) An example of extremely bad chosen landmarks.



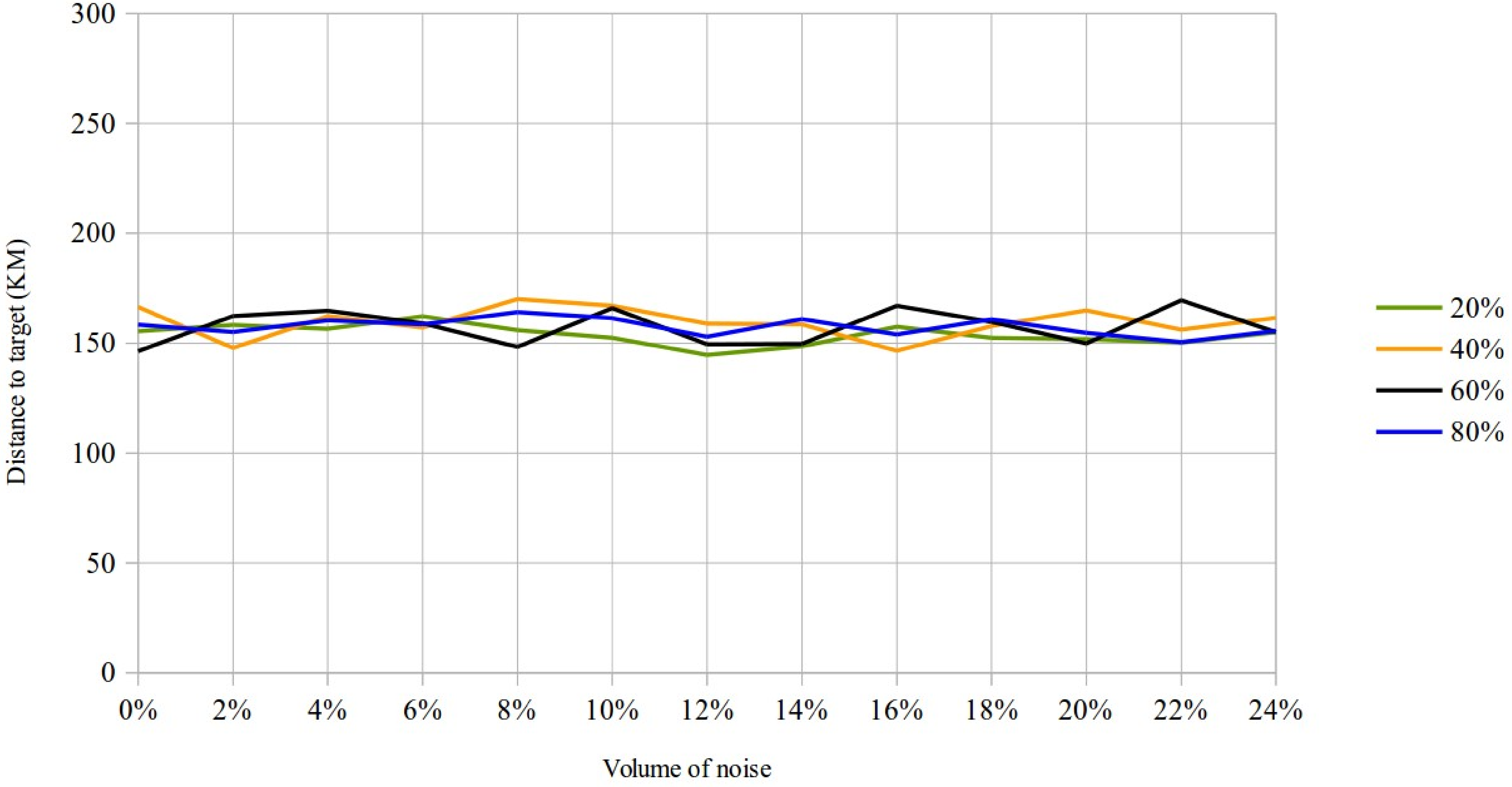
(b) An example of desirable chosen landmarks.

Factors impacting on the accuracy

$$Acc \propto \frac{P \times C}{F} - \|\frac{d}{dR}f(R)\|$$

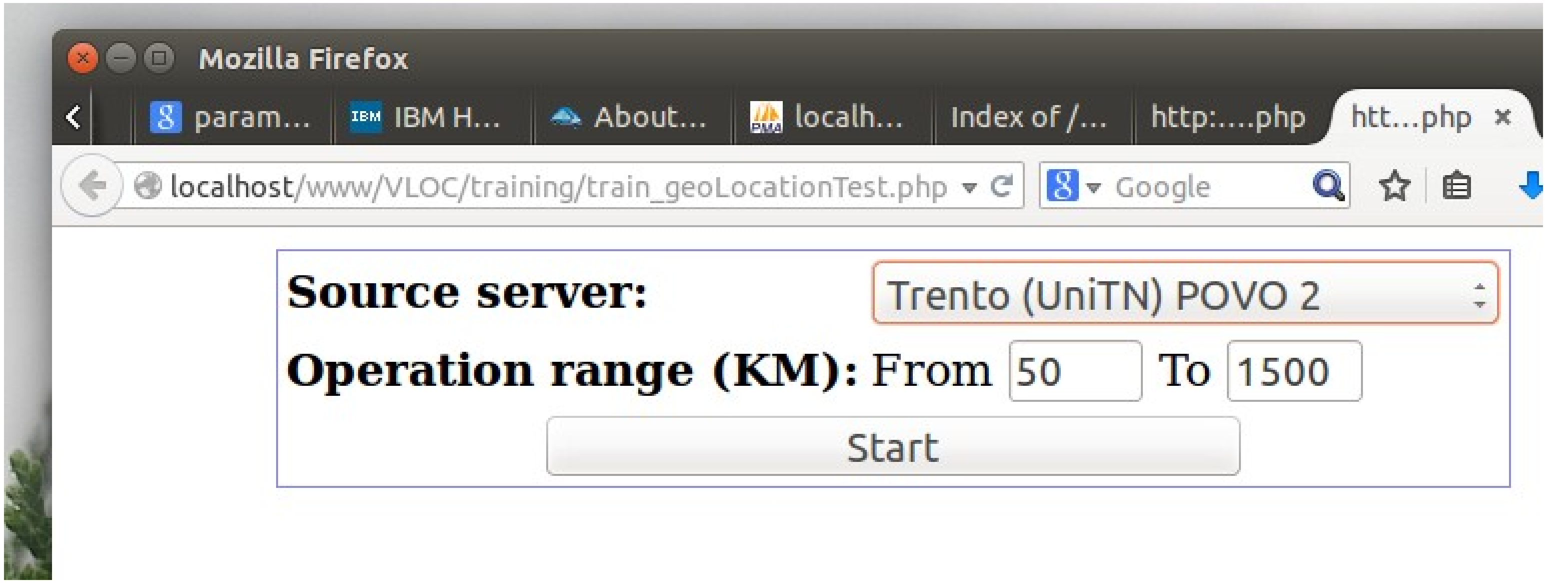
- ***Acc*** is the accuracy
- ***P*** is the frequency of latency measurement
- ***C*** is the confidence factor
- ***F*** refers to the network fluctuation
- ***f(R)*** is a function representing the changes of accuracy based on changes of range of operation

Imposing **delay** on network packets



Further Information:

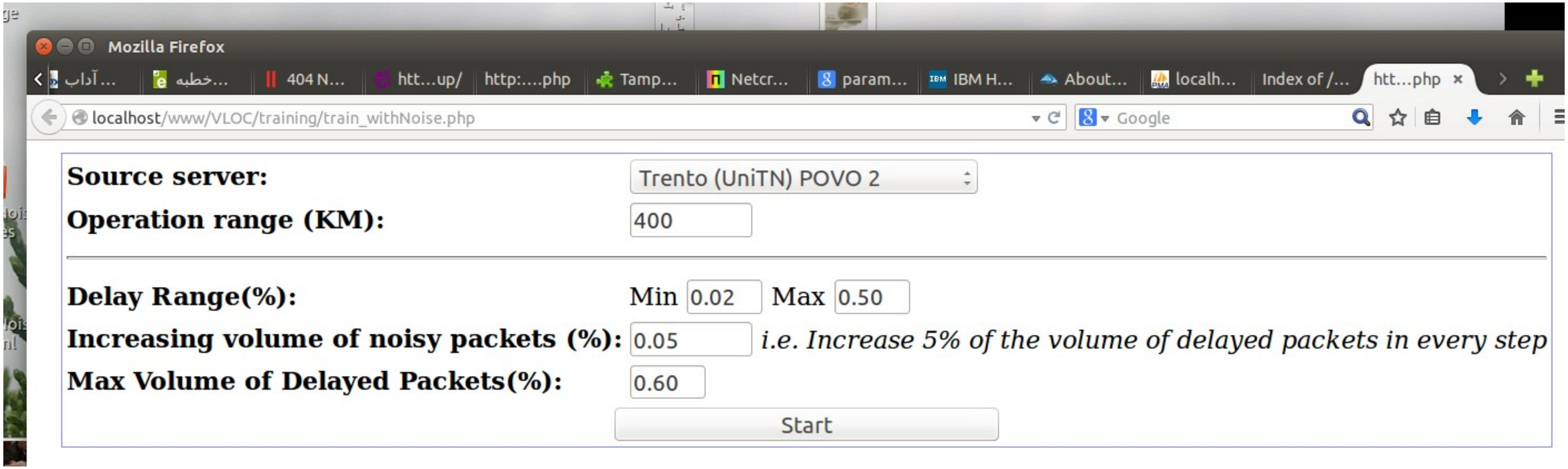
- No of collected URLs: ***187,439***
- No. of Observed Servers for Trento: ***17,264***
- Used ML technique: ***Polynomial Regression***
- Interface: ***Web***
- Implemented in: ***PHP/ MySQL*** *running on* ***Apache/Ubuntu***



localhost/www/VLOC/training/train_geoLocationTest.php

Source server: Trento (UniTN) POVO 2

Operation range (KM): From 50 To 1500



localhost/www/VLOC/training/train_withNoise.php

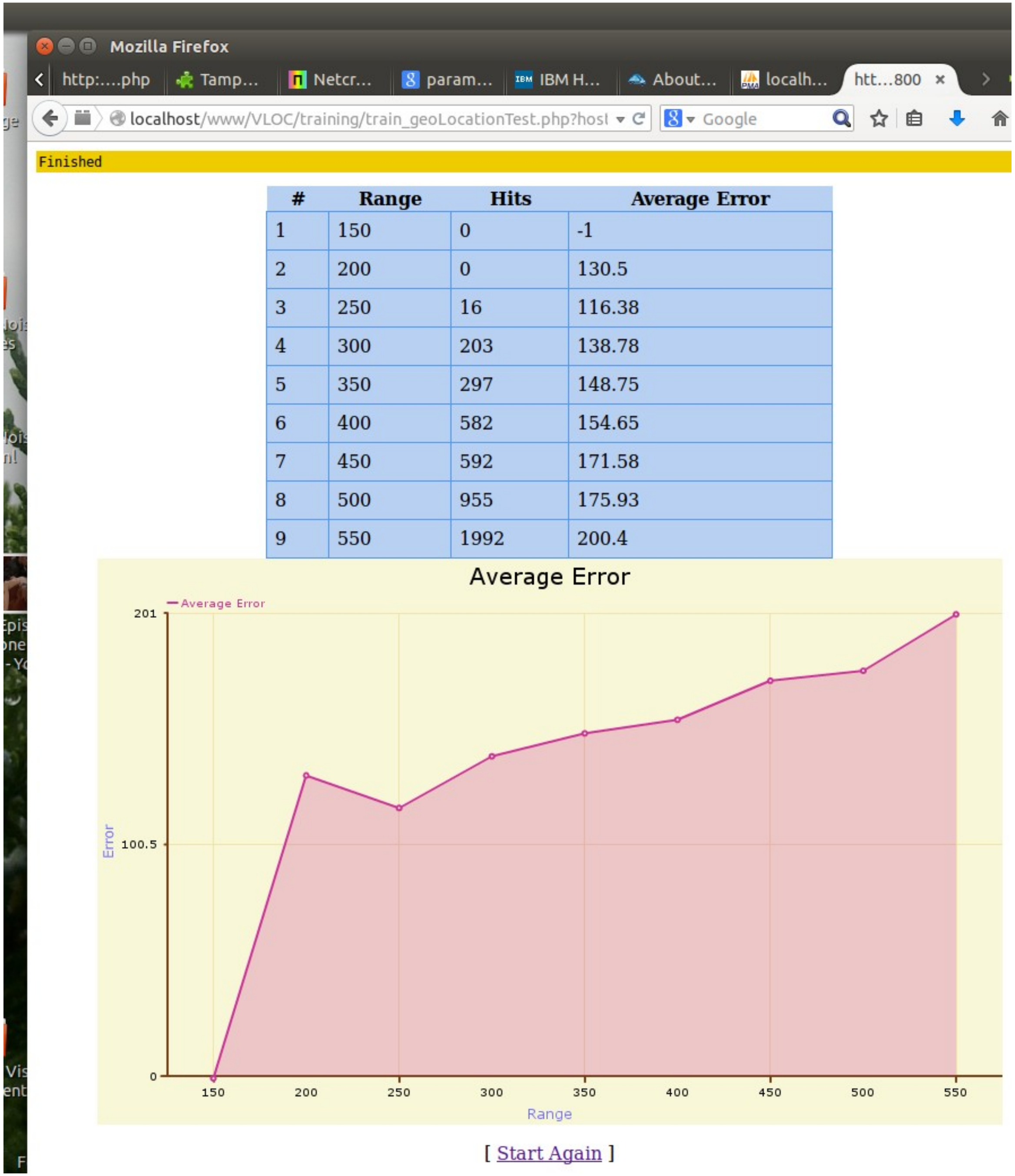
Source server: Trento (UniTN) POVO 2

Operation range (KM): 400

Delay Range(%): Min 0.02 Max 0.50

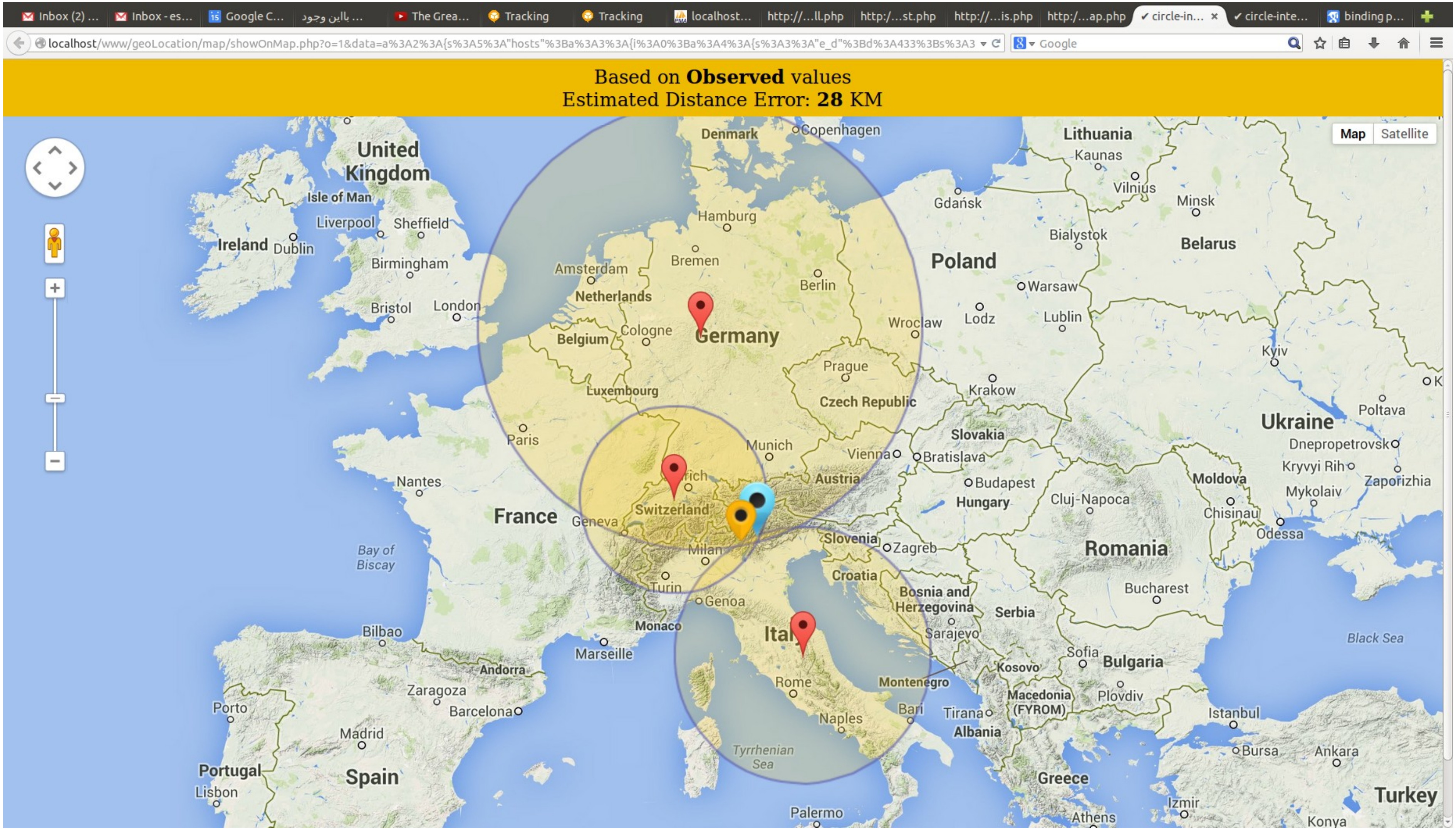
Increasing volume of noisy packets (%): 0.05 *i.e. Increase 5% of the volume of delayed packets in every step*

Max Volume of Delayed Packets(%): 0.60



#	Range	Hits	Average Error
1	150	0	-1
2	200	0	130.5
3	250	16	116.38
4	300	203	138.78
5	350	297	148.75
6	400	582	154.65
7	450	592	171.58
8	500	955	175.93
9	550	1992	200.4

[Start Again]



Based on **Observed** values
Estimated Distance Error: **28** KM