Standardized Localization Interface

Application-centric location information provisioning

Ivan Azcarate
(on behalf of Filip Lemic)

GeoIT WhereCamp'17, Berlin

Telecommunication Networks Group
Technische Universität Berlin

Motivation

- A variety of localization services exists:
  - Deployed in different environments (with overlaps).
- The aim should be seamless, accurate, and robust location information provisioning.
- Requirements for provisioning quality vary across applications;

Goals and contributions

- Goals:
  - **Seamless handover and fusion** of localization services;
  - **Straightforward addition** of new localization services;
  - In an **unified way** enabling portability of applications;

- We contribute with:
  - A middle-ware localization service architecture;
  - A proposal for standardized interaction (APIs);
  - A prototypical implementation of such an architecture;

Example of usage – WiFi horizontal handover

1. RSS decrease → an 'app' is triggered → the app requests location:

request_location('2D', '1m accuracy', '1s', 'movement')

request_context('map')

2. The user is still far from handover locations → but, the user is moving → request location again for the case the user moves for more than 3 m:

request_location(' '2D', '2m', '1s', 'on event – 3m', duration – 30sec)

3. The user is close to a handover location → request provisioning:

request location(' '2D', '1m', '0.5s', 'periodically', 'duration - 30sec')

4. The handover occurred → check if the user is progressing towards locations where no handover is needed:

request_location('movement')

5. Yes, s/he is! → we don't need location information for a while.

*Take home: heterogeneous requirements for location information (e.g. once/ periodically/on event, flexible accuracy/latency/duration, …)

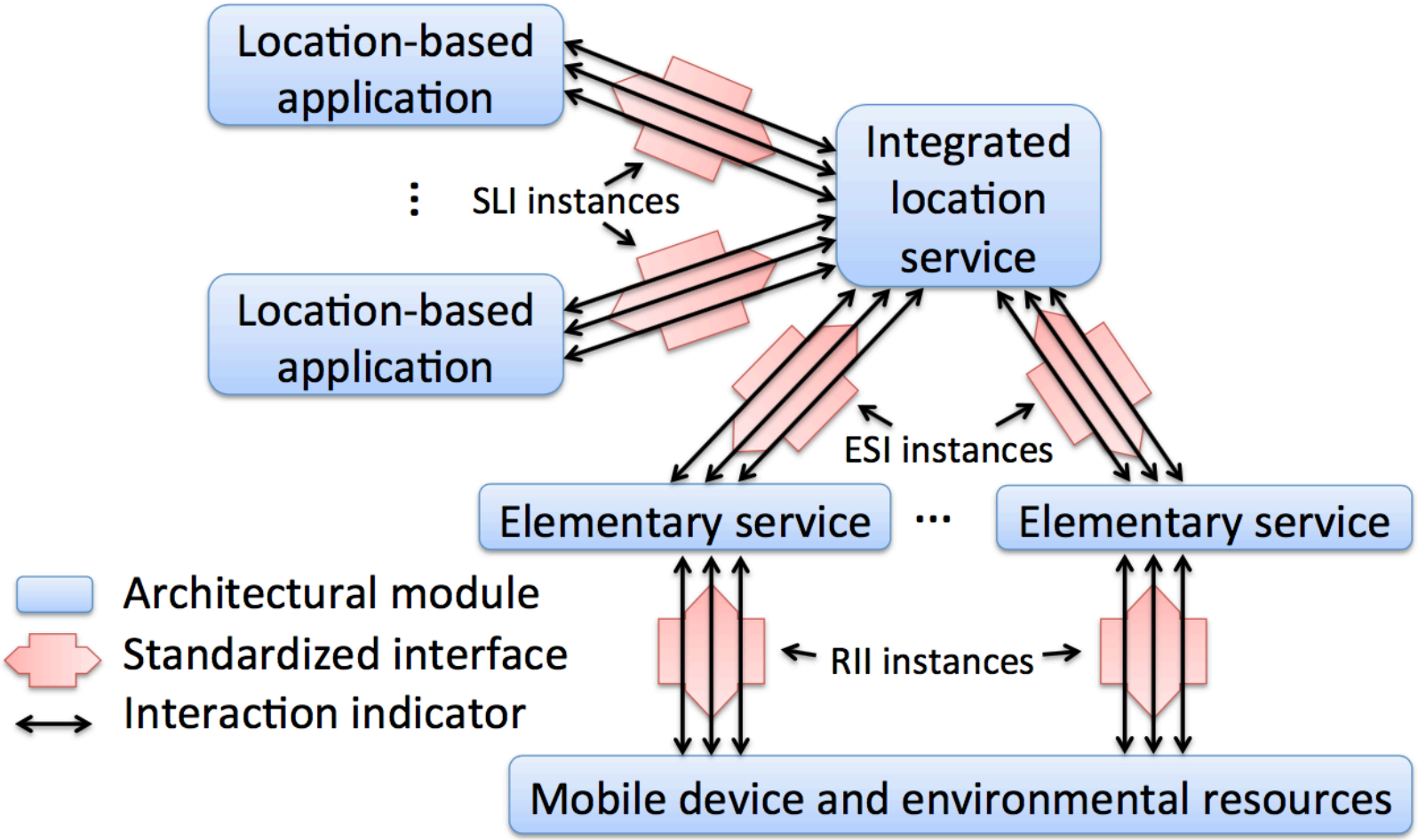
Standardized Localization Interface (SLI)

- Capturing the functionalities of the state of the art services;

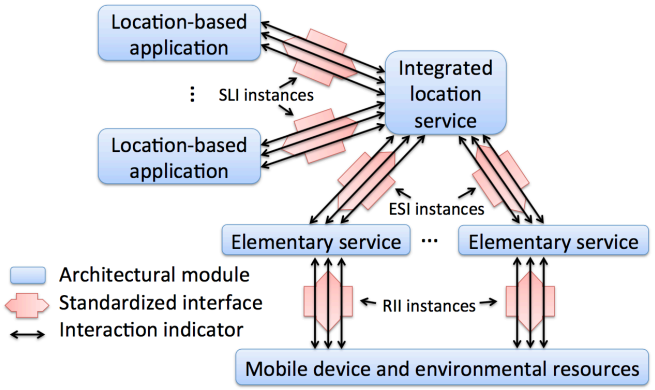
Action	Description	Direction
Specify policy	Specify trade-off policy	LA→IL
Request location	Request location information	LA→IL
Report location	Report location information	IL→LA
Request renewal	Request provisioning renewal	LA→IL
Request context	Request context of location information	LA→IL
Report context	Report context of location information	IL→LA

- Request location: **type, dimensionality, accuracy, period, on event, step, duration, movement**;
- Request context: **map (zero-point, map vs. physical sizes) or location types translation**;

Standardized modular localization service



Standardized modular localization service

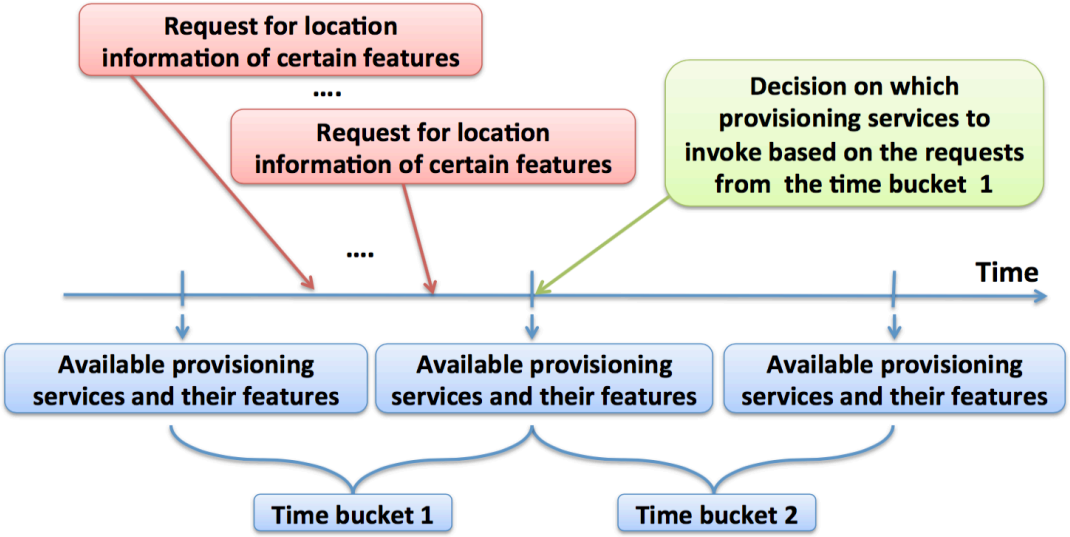


- Integrated location service:
  - Supplies and requests location information;
  - Manages the selection of services to be invoked;
  - A setting for fusion and caching of location information;
  - A setting for calculating location-context parameters;

- Interfaces:
  - Standardized Localization Interface (SLI);
  - Elementary Service Interface (ESI);
  - Resource Interaction Interface (RII);

Dynamic vs. time-bucketed operation

- Dynamic vs. bucketed selection decision?
  - High dynamicity that will further increase → frequent need for a new decision;
  - Dynamic selection has to be fast → algorithms have to be simple → not optimal for optimization;
- Time-bucketing is therefore selected:



Algorithms for selection of provisioning services

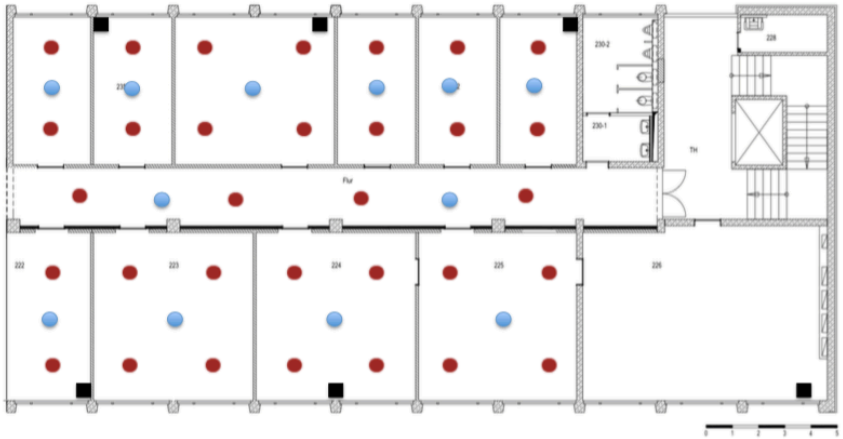
- Input:
  - Requests (accuracy, latency requirements) from the applications;
  - Provisioning features (accuracy, latency, power consumption);
- Objective:
  - Decision on which provisioning services to invoke;
- Subject to:
  - Fulfilling the latency (and subsequently accuracy) requirements from the applications, while:
    - PRSA → per-request minimizing power consumption;
    - PTSA → per time-bucket minimizing power consumption;

GDP-based implementation

- The GDP (Global Data Plane):
  - Natively supports a **single-writer log**-based messaging;
  - Provides a **publish/subscribe and REST** interfaces;
  - Logs are encrypted ensuring **data privacy and security**;
- Integrated Location Service:
  - Python 2.7-based multi-threaded daemon service;
- Publicly available on GitHub → **SLSR: Standardized Localization Service;**

Instantiation and evaluation setup

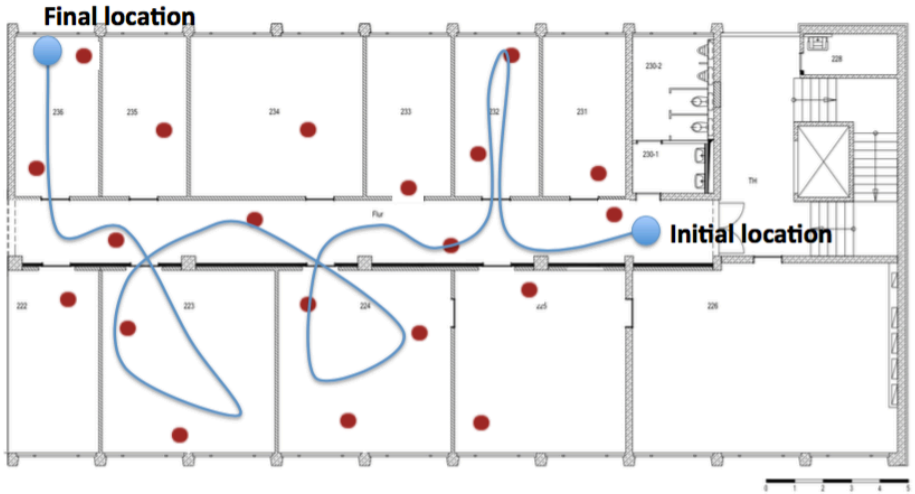
- Instantiation of a set of fingerprinting-based services;
- Localization scenario → to derive the expected performance of the instantiated provisioning services;



Provisioning service	Accuracy	Room accuracy	Latency	Power consumption
Euclidean small	3.14 m	52.5 %	0.66 s	2.0
Quantile small	3.05 m	55.0 %	0.72 s	2.0
Euclidean medium	2.41 m	62.5 %	0.86 s	2.0
Quantile medium	2.22 m	65.0 %	0.91 s	2.0
Euclidean large	1.96 m	75.0 %	1.05 s	2.0
Euclidean semantic	/	52.5 %	0.66 s	1.0
Quantile semantic	/	55.0 %	0.72 s	1.0

Evaluation

- Tracking scenario:
  - To demonstrate the benefits of different functional components;



#	Accuracy	Latency	Provisioning type	Location type
1	1.5 m	2.0 s	periodic	local
2	100 %	1.5 s	periodic	semantic
3	100 %	5.0 s	periodic	semantic
4	100 %/1.5 m	3.0 s	periodic / on event (1 m)	semantic / local
5	1.0 m	1.0 s	periodic	local
6	1.5 m	1.0 s	periodic / on event (3 m)	local
7	2.0 m	1.5 s	on event (1 m)	local

Evaluation results

- Basic – basic functionalities of the integrated location service;
- Caching & mapping – caching/mapping functionalities introduced;
- Long-term interpretation – pushing the intelligence to the ILS;
- Dynamic – "God-view" on provisioning features;

Type	Total number of requirements	Accuracy satisfaction	Latency satisfaction	Both requirements satisfaction	Consumed power
Per-Request Satisfaction Algorithm (PRSA)					
Basic	659	141	388	113	2021
Caching & mapping	659	352	504	303	1201
Long-term interpretation	659	361	531	313	1167
Dynamic	659	451	528	347	1387
Per-Time-Bucket Satisfaction Algorithm (PTSA)					
Basic	659	98	388	68	1510
Caching & mapping	659	331	504	281	854
Long-term interpretation	659	341	531	294	833
Dynamic	659	433	528	339	1041

Results

- Global optimization consumes ~25% less power;
- Accuracy satisfaction of global optimization is ~25% smaller;
- Mapping and caching capabilities reduce total consumed power and benefit both accuracy and latency;
- Pushing “the intelligence” to the ILS benefits latency and accuracy;
- “God-view” improves accuracy satisfaction;

Conclusions

- Each of the defined functional components benefits the overall performance of the SLSR.
- In tracking scenarios, there is a dependence between accuracy and latency.
- Satisfaction of requirements from the applications is more important than “provisioning on steroids”;
- Standardization is needed.

The end…

Thank you!

Ivan Azcarate

Telecommunication Networks Group, TU Berlin

azcarate@tkn.tu-berlin.de
lemic@tkn.tu-berlin.de

Support for this work has come from the EU Projects EVARILOS (grant no. 317989) and eWINE (grant no. 688116), the German Academic Exchange Service (DAAD), the UC Berkeley Swarm Lab and the Berkeley Wireless Research Center (BWRC). This work was also supported in part by TerraSwarm, one of six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA.