TKN Telecommunication Networks Group

Technical University Berlin

Telecommunication Networks Group

Service for Calculation of Performance Metrics of Indoor Localization Benchmarking Experiments

Filip Lemic

lemic@tkn.tu-berlin.de

Berlin, July 2014

TKN Technical Report TKN-14-003

TKN Technical Reports Series

Editor: Prof. Dr.-Ing. Adam Wolisz

Abstract: This report presents the service for calculation of performance metrics of indoor localization benchmarking experiments. By performance metrics we mean point and room level accuracy of location estimation, response time and power consumption of the system under test. The service is developed in order to standardize the way performance metrics of indoor localization benchmarking experiments are calculated, to automatize the storage of metrics into specific storage service and to remove the burden of manual calculation of metrics, removing the possibility of the errors in calculations. The service provides two ways of usage, namely *offline* and *online* calculation of performance metrics. *Offline* calculation can be used for calculating performance metrics of a previously performed experiment, so on the set of ground truths and estimates. Contrary to that, *online* calculation calculates and updates metrics in a real-time, so ground truths and corresponding estimates can be provided sequentially to the service and the service will take care of calculation and updating the performance metrics in the database. The services is publicly available and easy to use only by defining a specific message and sending it on a provided HTTP URL.

Keywords: indoor localization, benchmarking, performance evaluation, metrics calculation, cloud service

Acknowledgments: This work has been partially funded by the European Commission (FP7-ICT-FIRE) within the project EVARILOS (grant No. 317989).

Contents

Chapter 1

Introduction

Indoor localization, in sense of estimating location in the environments where GPS (Global Positioning System) usually fails, i.e. indoors, is highly needed and important service in numerous applications. However, obtaining the accurate, fast and energy efficient indoor localization service is a highly demanding and challenging task. Many research efforts have been taken in improving the accuracy, delay, power consumption, etc. of indoor localization solutions and various results have been presented. However, there is no standardized way of obtaining and presenting the performance results of indoor localization benchmarking experiments. Namely, just as the example, some results present average localization error as the performance result of their systems [1], while others present median [2] or 80 percentile error [3]. We see two issues raising from that fact. First, the results presented in that way are incomparable and not prone to "simple" errors in calculations that can occur. Secondly, in our opinion, presenting the metric only partially, such as average localization error, is not enough to characterize the performance of indoor localization solutions.

In these facts we find the motivation for our work. We present a service for calculation of an extensive set of metrics for characterizing the performance of indoor localization. In our previous work [4] we have detected point and room accuracy, delay of indoor localization and power consumption as important performance metrics of indoor localization. Note that these are only performance metrics, since others like cost or deployment efforts are also important, but out of the score of this work. Our service is publicly available for calculation of an extensive statistical information regarding the performance of indoor localization in two manners: *offline* and *online*. By performing an *offline* calculation we assume "feeding" the service with a set of evaluation points and estimates, while *online* calculation assumes inputing evaluation points and estimates one-by-one, where the service each time reports (and updates) the statistical information about the performance of the evaluated solution.

The service is tightly connected to our previous work [5], services for storing the data used in indoor localization and benchmarked results of indoor localization, meaning that these services can be combined into calculation of performance metrics which we describe in this report and storage of raw data and performance metrics which is described in the previous work. By combing these services we aim on simplifying their usage, namely providing to the user only one well defined interface that can easily be used for storage of raw data of indoor localization and calculation and storage of the results of indoor localization benchmarking experiments.

The rest of the report is structured as follows. Chapter II of the report presents the design of the service for calculation of performance metrics of indoor localization experiments. We present two main functions of the service, namely *offline* and *online* calculation of the performance metrics of indoor localization benchmarking experiments. Chapter III gives the implementation of these functions using RPC (Remote Procedure Call) API and Protocol Buffer messages. Finally, Chapter IV concludes the report.

Chapter 2

Service Design

As mentioned in the introduction of the report, the idea of the service is to support two types of calculation of performance metrics on indoor localization benchmarking experiments, namely *offline* and *online* calculation.

2.1 *Offline* Calculation of Performance Metrics

The general idea of *offline* calculation of metrics is presented in Figure 2.1. After collecting the set of location estimates from the evaluated indoor localization solution and corresponding ground truth locations where the measurements were taken, user is able to input these data into the service, and the service will output the set of statistical information describing the accuracy of the localization solution. Additionally, user can measure the latency (also known as response time or delay) or power consumption of location estimation at each location and if this kind of data is given as an input to the service, the service will provide statistics regarding response time or / and power consumption of the indoor localization solution. The full description of what "statistical information about the performance of indoor localization actually" means, i.e. what is the output of the service, will be given in the following chapter of the report. Optionally, user is able to define if the raw data that the localization solution is using should be requested and stored. The benefits of storing the raw data are presented in [5]. Finally, user can optionally define if and where the results (ground truth + estimates, overall statistical information, optionally latencies and power consumption) should be stored. More information about the format of stored results can be also found in [5].

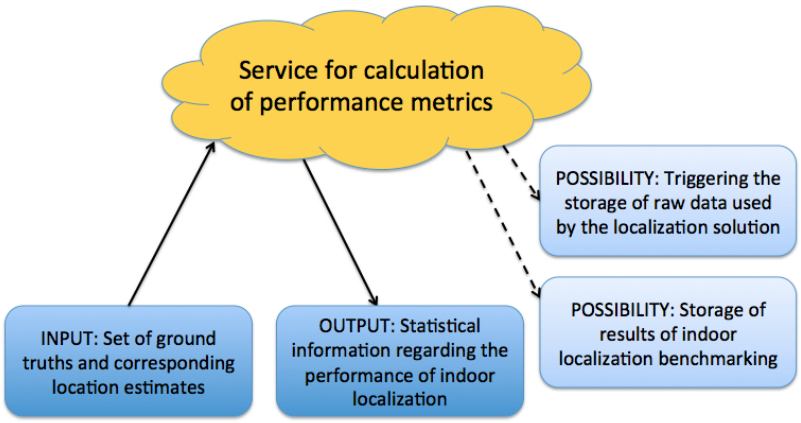

Figure 2.1: *Offline* calculation of statistical information of indoor localization benchmarking

2.2 *Online* Calculation of Performance Metrics

Online calculation of performance metrics is similar to the *offline* in the sense that it provides the same output (Figure 2.2). However, the idea here is to sequentially provide the ground truth location and corresponding estimate given by the indoor localization solution, define where the performance metrics should be stored and the service will automatically update the previously existing evaluation data (if any), store the data including the new evaluation point and report the results. Alternatively, service can also be used for interfacing the indoor localization solution (SUT in Figure 2.2), where the service can, given only the ground truth where the localization solution currently is, capture the estimate and latency of location estimation (and the power consumption if the localization solution provides the interface for accessing this information). Then, in the same way, service can update the currently stored data with the new evaluation point and report the results. More about interfacing the indoor localization solution will follow in the next chapter.

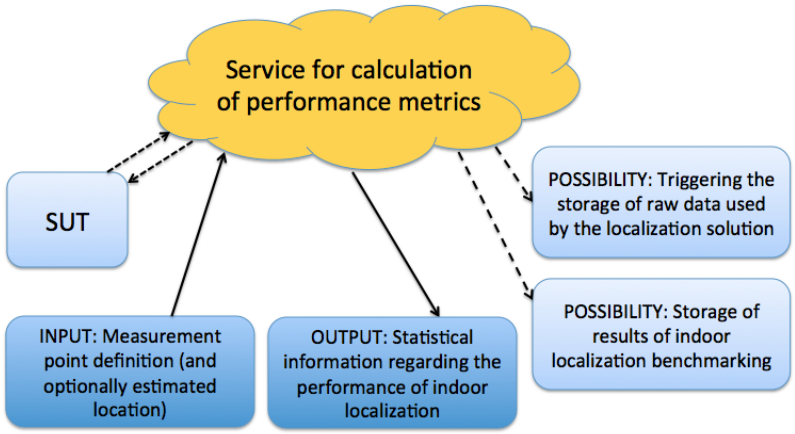

Figure 2.2: *Online* calculation of statistical information of indoor localization benchmarking

2.3 Design Advantages for Service Users

The most important advantage of the described design of the service for calculation of performance metrics of indoor localization benchmarking experiments is giving the user various possibilities of usage. Namely, user is able to perform the whole experiment for benchmarking of indoor localization solution, simply feed the data to the service and the service will report the results. Furthermore, user can continuously feed evaluation points and estimates to the service and the service will automatically take care of updating the statistical information and storing the results. Finally, user is able to just define the localization of the indoor localization solution's interface and if the interface is properly defined the service can take care of interfacing the solution, triggering the solution, capturing the output, calculating the metrics and storing the data.

Apart from the described advantage, the service can be considered as a "step forward" in the standardization of metrics that should be used for characterization of indoor localization's performance. Finally, this service reduces the load for the users, namely it takes care of both metrics calculation and storage with only one well defined request.

Chapter 3

Service Implementation

The implementation of the service is relatively simplistic, consisting of two RPC (Remote Procedure Call) functions over HTTP protocol. For each call a different Protocol Buffer [6] message has to be defined and send on specific URL using HTTP POST request. Service is developed in Python 2.7. programming language [7] using the Flask framework [8], and is running as the EC2 [9] instance in Amazon cloud [10]. Two different messages and calls correspond to *offline* and *online* calculation of metrics, as described previously. The service is currently available for public usage on the following URL:

`http://ebp.evarilos.eu:5010/`

3.1 Service Input

As said before, service can calculate and store the performance metrics of indoor localization benchmarking experiments in *offline* and *online* manner. For this purpose two different RPC functions can be used, with two different Protocol Buffer messages that should be send as an input to the service. Two different types of messages are in details described in the following section.

3.1.1 Request for *Offline* Calculation of Performance Metrics

For the *offline* calculation of indoor localization performance metrics the Protocol Buffer message given bellow should be used. Firstly, the message defines the time when the metrics are calculated and the scenario in which the experiment was performed. Scenario is defined with various message fields, describing the testbed, interference scenario, and system under test (SUT that is being evaluated. Secondly, the Protocol Buffer message consists of a number of evaluation points (i.e. locations) in which the SUT has been evaluated. Each location consists of the ground truth coordinates (x,y,z) with the corresponding room label, estimates coordinates (x,y,z), also with the corresponding estimated room label, latency of location estimation at a given evaluation point and power consumption at a given point. Power consumption can be also measured in the whole experiment (contrary to measurement per evaluation point due to complexity of capturing this metric) and a specific message field for that purpose is defined. Finally, the message consists of a set of "control" fields, which define if the calculated metrics should be stored and where. Naturally, all fields of the message don't necessarily have to be defined, and these fields are defined with the parameter "optional" before the type definition, as shown in the message below.

```
message ece_type1 {
```

```
  required int64 timestamp_utc = 1; // Timestamp - milliseconds from 1.1.1970
  required string experiment_label = 2; // Name of experiment
  repeated Evaluation_point locations = 3; // Measurement locations
  required Scenario_description scenario = 4; // Describes a scenario
  optional double power_consumption_per_experiment = 5; // Power consumption in
  // experiment
  optional bool store_metrics = 6 [default = false]; // Storing metrics?
  optional string metrics_storage_URI = 7; // URI for storing processed data
  optional string metrics_storage_database = 8; // Name of database for storing
  optional string metrics_storage_collection = 9; // Name of collection for storing

  message Evaluation_point {
    required int32 point_id = 1; // ID of each point in experiment
    optional int32 localized_node_id = 2; // ID of localized node
    required double true_coordinate_x = 3; // Ground truth - X-coordinate
    required double true_coordinate_y = 4; // Ground truth - Y-coordinate
    optional double true_coordinate_z = 5; // Ground truth - Z-coordinate
    optional string true_room_label = 6;  // Ground truth - room label
    required double est_coordinate_x = 7; // Estimated location - x-coordinate
    required double est_coordinate_y = 8; // Estimated location - y-coordinate
    optional double est_coordinate_z = 9; // Estimated location - z-coordinate
    optional string est_room_label = 10; // Estimated location - room label
    optional double latency = 11; // Latency of location estimation
    optional double power_consumption = 12; // Power consumption for this location
  }

  message Scenario_description {
    required string testbed_label = 1; // Testbed label
    required string testbed_description = 2; // Testbed description
    required string experiment_description = 3; // Experiment description
    required string sut_description = 4; // SUT description
    required string receiver_description = 5; // Receiver description
    required string sender_description = 6; // Sender description
    required string interference_description = 7; // Interference description
  }
}
```

3.1.2 Request for *Online* Calculation of Performance Metrics

Second message that can be defined and used as an input to the service is used for *online* calculation of performance metrics, meaning that evaluation points can be inputed to the service sequentially. This given the possibility of calculating performance metrics in real-time which can be used for example for automation of indoor localization benchmarking experiments.

This Protocol Buffer message is similar to the previous one in the sense that it defines the time and the scenario in the same way. However, instead of a set of locations, this message defines separately ground-truth and estimate of only one location. Ground truth is defined with the coordinates (x, y, z) and the corresponding room. Location estimate is defined with the estimated coordinates (x, y, z), estimated room label, delay of location estimation and power consumption of a SUT device used for estimating the location.

```
message ece_type2 {

  required int64 timestamp_utc = 1; // Timestamp - milliseconds from 1.1.1970
```

```
required string experiment_label = 2; // Name of experiment
required Scenario_description scenario = 3; // Describes scenario
required bool request_raw_data = 4 [default = false]; // Raw data requested
// in experiment?
required bool request_estimates = 5 [default = false]; // Location estimates
// requested from SUT?
required bool store_metrics = 6 [default = false]; // Metrics stored in
// experiment?
required bool request_power_consumption = 7 [default = false]; // Power
// estimate requested from SUT?
required Ground_truth ground_truth = 8; // Ground truth location
optional Estimated_location estimate = 9; // Estimated location

// Interfacing to system under test

optional string sut_location_estimate_URI = 10; // URI of SUT's interface for
// requesting location estimate
optional string sut_raw_data_URI = 11; // URI of SUT's interface for requesting
// to store raw data
optional string sut_power_estimate_URI = 12; // URI of SUT's interface for
requesting power estimate

// Storing the processed data

optional string metrics_storage_URI = 13; // URI for storing processed data
optional string metrics_storage_database = 14; // Name of database for
// storing the results
optional string metrics_storage_collection = 15; // Name of collection for
// storing results

message Ground_truth {
  required int32 point_id = 1; // ID of each point in experiment
  required int32 localized_node_id = 2; // ID of localized node
  optional string point_label = 3; // Ground truth - point label
  optional double true_coordinate_x = 4; // Ground truth - X-coordinate
  optional double true_coordinate_y = 5; // Ground truth - Y-coordinate
  optional double true_coordinate_z = 6; // Ground truth - Z-coordinate
  optional string true_room_label = 7; // Ground truth - room label
}

message Estimated_location {
  optional double est_coordinate_x = 1; // Estimated location - X-coordinate
  optional double est_coordinate_y = 2; // Estimated location - Y-coordinate
  optional double est_coordinate_z = 3; // Estimated location - Z-coordinate
  optional string est_room_label = 4; // Estimated location - room label
  optional double latency = 5; // Latency of location estimation in milliseconds
  optional double power_consumption = 6; // Power consumption of location
  // estimation in milliwatts
}

message Scenario_description {
  required string testbed_label = 1; // Testbed label
  required string testbed_description = 2; // Testbed description
  required string experiment_description = 3; // Experiment description
  required string sut_description = 4; // SUT description
```

```
    required string receiver_description = 5; // Receiver description
    required string sender_description = 6; // Sender description
    required string interference_description = 7; // Interference description
  }
}
```

They are two possibilities for using this message for calculation of *online* metrics. Firstly, the message can be used without interfacing the SUT, where the location estimate is provided in the message itself. This can be achieved by setting the control field *request_estimates* to "False". Opposite to this usage possibility, service can also be used for interfacing the SUT. For this purpose SUT has to provide an HTTP URI on which their algorithm listens for requests for location estimation. Upon request, the algorithms must be able to provide the location estimate as a JSON response in the following format:

```
{
  "coordinate_x":   'Estimated location: coordinate x',
  "coordinate_y":   'Estimated location: coordinate y',
  "coordinate_z":   'Estimated location: coordinate z',
  "room_label":     'Estimated location: room'
}
```

JSON parameters *coordinate_x* and *coordinate_y* are required parameters and as such they must be reported upon request. Parameter *coordinate_z* is an optional parameter and if provided 3D localization error can be calculated. Finally, parameter *room_label* is an optional parameter providing the estimated room label.

Service can also trigger the collection of the raw data from the SUT by sending a plain HTTP GET request on an URI specified in the message field *sut_raw_data_URI*. If the user want to trigger the storage of the raw data the parameter *request_raw_data* has to be set on "True". Note that service can only trigger the collection of raw data of indoor localization benchmarking experiment, but the SUT itself has to be accommodated for storage of the data. Finally, SUT has to provide an interface that can or request provide an estimate of power consumption of the SUT. Interface needs to provide an URI (message parameter *sut_power_estimate_URI*) that accepts plain HTTP GET requests and replies with a number representing an estimate of power consumption of previous location estimation performed by the SUT.

Similar as the message for *offline* calculation of performance metrics, this message has a set of "control" fields. Using those fields user can define if the raw data collection should be triggered, if the performance metrics should be stored and if location estimates should be requested from the SUT. Finally, only the "required" fields of the message have to be defined, while "optional" fields can be empty without causing any errors in the service.

3.2 Service Output

After triggering the service with one of the previously defined input messages, the service will reply with an output message defined below. This message can also easily be stored into the database for storing the benchmarking results of indoor localization by setting properly the parameters of the input message as described in the text above. The output message consists of a time when the metrics were calculated, i.e. when the experiment was performed, and the scenario in which the experiment was

performed, namely testbed description, SUT description and interference description. Furthermore, the message consists of a set of evaluation points, described with IDs, ground truth and estimated coordinates (x, y, z) and room labels, latencies of location estimation and power consumptions. In addition to that, for each evaluation point the 2D and 3D localization error is calculated with the binary statement of the correctness of the estimated room.

Finally, the output message consists of a set of fields defined in *primary_metrics* part of the message, used for describing the overall performance of indoor localization solution in an experiment. This set of fields gives various statistical information (average, median, root-mean-square (RMS), 75 percentile, 90 percentile, minimum, maximum) of different metrics. Namely, the metric are point level accuracy of indoor localization, defined as Euclidean distance between ground truth and estimated coordinate, room level accuracy defined as correctness of estimated room, latency as the time needed to estimate the location and power consumption of the SUT.

```
message Experiment {
  required int64 timestamp_utc = 1; // Timestamp - milliseconds from 1.1.1970
  required string experiment_label = 2; // Name of experiment
  repeated Measurement_point locations = 3; // Each experiment consists of a
  // number of locations
  required Scenario_description scenario = 4; // Describes scenario
  required Primary_metrics primary_metrics = 5; // Calculation of primary metrics

  message Measurement_point {
    required int32 point_id = 1; // ID of each point in experiment
    optional int32 localized_node_id = 2; // ID of localized node
    optional string point_label = 3; // Ground-truth: point label
    required double true_coordinate_x = 4; // Ground-truth: X-coordinate
    required double true_coordinate_y = 5; // Ground-truth: Y-coordinate
    optional double true_coordinate_z = 6; // Ground-truth: Z-coordinate
    optional string true_room_label = 7; // Ground-truth: room
    optional string est_point_label = 8; // Estimated location: point label
    required double est_coordinate_x = 9; // Estimated location: X-coordinate
    required double est_coordinate_y = 10; // Estimated location: Y-coordinate
    optional double est_coordinate_z = 11; // Estimated location: Z-coordinate
    optional string est_room_label = 12; // Estimated location: room
    optional double latency = 13; // Latency of location estimation at this location
    optional double localization_error_2D = 14; // 2D localization error of each
    // point in experiment
    optional double localization_error_3D = 15; // 3D localization error of each
    // point in experiment
    optional double localization_correct_room = 16; // Room error of each point
    // in experiment
    optional double power_consumption = 17; // Power consumption estimate for each
    // point in experiment
  }

  message Primary_metrics {
    optional double error_2D_average = 1; // Average 2D error of accuracy of all
    // points in experiment
    optional double error_2D_median = 2; // Median 2D error of accuracy of all
    // points in experiment
    optional double error_2D_std = 3; // 2D error standard deviation of accuracy of
    // all points in experiment
    optional double error_2D_min = 4; // Min 2D error of accuracy of all points
```

```
    // in experiment
    optional double error_2D_max = 5; // Max 2D error of accuracy of all points
    // in experiment
    optional double error_3D_average = 6; // Average 3D error of accuracy of all
    // points in experiment
    optional double error_3D_median = 7; // Median 3D error of accuracy of all
    // points in experiment
    optional double error_3D_std = 8; // 3D error standard deviation of accuracy of
    // all points in experiment
    optional double error_3D_min = 9; // Min 3D error of accuracy of all points
    // in experiment
    optional double error_3D_max = 10; // Max 3D error of accuracy of all points
    // in experiment
    optional double room_error_average = 11; // Average room accuracy error
    optional double latency_average = 12; // Average latency
    optional double latency_median = 13; // Latency median
    optional double latency_std = 14; // Latency standard deviation
    optional double latency_min = 15; // Latency min
    optional double latency_max = 16; // Latency max
    optional double power_consumption_average = 17; // Average power consumption
    optional double power_consumption_std = 18; // Standard deviation of
    // power consumption
    optional double power_consumption_min = 19; // Min power consumption
    optional double power_consumption_max = 20; // Max power consumption
    optional double power_consumption_median = 21; // Median power consumption
  }

  message Scenario_description {
    required string testbed_label = 1; // Testbed label
    required string testbed_description = 2; // Testbed description
    required string experiment_description = 3; // Experiment description
    required string sut_description = 4; // System under test description
    required string receiver_description = 5; // Receiver description
    required string sender_description = 6; // Sender description
    required string interference_description = 7; // Interference description
  }
}
```

Chapter 4

Conclusion

In this report we have presented the publicly available service for calculation of the performance metrics of indoor localization benchmarking experiments. The service can be used in two ways, *offline*, providing the possibility of calculating the set of metrics on the previously performed experiment, and *online*, giving the possibility of calculating the metrics on the run, while the experiment is preformed. Service also provides possibility of automatically storing the metrics, triggering the collection of raw data or interfacing the SUT, which simplifies the execution of indoor localization benchmarking experiments and tries to standardize the set of metrics that are used for comparing the performance of different indoor localization systems.

Bibliography

[1] H. Wang et al: “No need to war-drive: unsupervised indoor localization”, Proceedings of the 10th international conference on Mobile systems, applications, and services. ACM, 2012.

[2] K. Chintalapudi, A. P. Iyer, V. N. Padmanabhan. “Indoor localization without the pain”, Proceedings of the sixteenth annual international conference on Mobile computing and networking. ACM, 2010.

[3] H. Lim, L.C. Kung, J.C. Hou, H. Luo: “Zero-configuration, robust indoor localization: Theory and experimentation”, 2005.

[4] T. Van Haute et al: “The EVARILOS Benchmarking Handbook: Evaluation of RF-based Indoor Localization Solutions”, EVARILOS Project, 2013.

[5] F. Lemic, V. Handziski: “Data Management Services for Evaluation of RF-based Indoor Localization”, TKN Technical Report Series, 2014.

[6] Buffers, Protocol: “Google’s Data Interchange Format”, 2011.

[7] “”Python Programming Language”: http://www.python.org/

[8] “Flask: Python Microframework”: http://www.pocoo.org/projects/flask/#flask/

[9] “Amazon Elastic Compute Cloud (Amazon EC2)”: http://aws.amazon.com/ec2/

[10] “Amazon Web Services (AWS)”: http://aws.amazon.com/