Supplementary Material

Additional explanations about computational architecture, requirements and software design

Requirements:

Figure 1 below summarizes the main needs of framework.

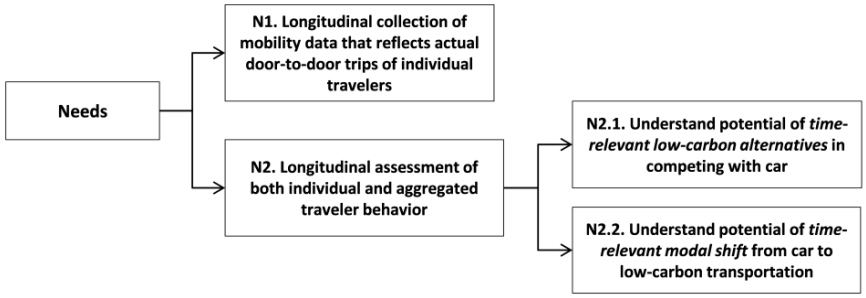

Figure 1. Needs related to analyzing travel behavior and exploring potential of time-relevant low-carbon alternatives.

In order to meet the above needs, the following framework requirements are defined in two interrelated groups A) Data collection, and B) Analysis of collected travel data. Defining system requirements and tracing the needs to requirements is a well-established software development practice [1]. **Error! Reference source not found.** summarizes traceability of framework needs to framework requirements, and from requirements to the software implementation. The requirements highlighted in **bold** are implemented in this paper.

Table 1: Needs, system requirements and its implementation.

Needs	Requirements	Implemented by
N1	RQ1.1.1: Identify trip legs and their travel paths	TrafficSense (TS) mobile app implemented by [2], [3]
	RQ1.1: Identify whole multimodal door-to-door trips	Section 3.3: Trip-extraction module implemented by this paper.
	RQ1.2: Trip to traveler assignment	
	RQ2: Resolution, spatial and temporal accuracy, and spatial and temporal coverage	TrafficSense (TS) mobile app implemented by [2], [3]
	RQ3: Automatic collection	
	RQ4: Anonymous collection	
N2	**RQ5: Identify time-relevant low-carbon alternatives**	Section 3.4: Time-relevant analysis method and post-processing implemented by this paper.
	RQ6: Identify modal shift potential	
	RQ7: Present spatial, temporal, per-trip and per-person viewpoints	Sections 3 and 4: Applying our method to travel data collected in Helsinki region.

Requirements in detail:

A) Data collection requirements

High resolution mobility data at the level of individual traveler is required to achieve a more realistic estimation of potential for time-relevant low-carbon travel. At the moment, no mobility data source is available for Helsinki region that satisfies our data needs. RQ1 denotes the items that should be directly available or otherwise extractable from the dataset records. RQ1.1.1 denotes that for each door-to-door trip, the system requires individual trip sections (trip-legs) that make that trip, including all chosen transportation modes during the trip. RQ1.2 denotes that each detected door-to-door trip should be attributed to the individual traveler that made the trip. RQ2 denotes the attributes of the data items. RQ2.1 denotes that time-relevant analysis requires data on door-to-door trips at the level of individual traveler. RQ2.2 denotes the required spatial accuracy of OD geolocation points. RQ2.3 denotes that we require accurate-enough timestamps of start-time and end-time of each trip. Having more accurate timestamps is particularly important when computing PT choices. In situations where PT is not frequent, a small time deviation may make the traveler miss the opportunity of traveling by PT. RQ2.4 denotes a wide spatial coverage, ideally, by having data of trips throughout the whole city region. RQ2.5 denotes a long-enough temporal coverage, ideally, a dataset collected over several months and representing mobility activities during the whole day and all days of the week. RQ3 denotes that data collection should be as automatic as possible unlike conventional surveys where travelers have to manually record all details of their trips. RQ4 considers privacy of travelers. Although according to RQ1.2, each trip is assigned to one traveler, traveler should be known only by an anonymous unique ID.

B) Analysis requirements

These requirements denote properties of the software system for analyzing the collected travel behavior dataset. RQ5.1 denotes that such software system should meet two main conditions to be effective in time-relevant analysis: Firstly (RQ5.1.1), system should consider both travel-time and emission factors, targeting for low-carbon mobility but at the same time considering traveler's limited travel-time budget. Traveling with the computed alternative modes should be low-carbon but also affordable in terms of travel-time to address the key role of time in travel mode choice. Secondly (RQ5.1.2), each computed alternative route plan should relate to one observed door-to-door multimodal trip [4]. Not the individual legs but the whole door-to-door trip should be substituted with low-carbon alternative mode(s) and route. RQ5.2 refers to the need to utilize or implement a trip computation method. To compute alternative trips with different modes of transportation, we need to make use of the following external data: PT route and

schedule, and road network of city. RQ7 denotes that the system should summarize the computed potentials from different spatial, temporal, per-trip and per-traveler viewpoints.

Figure 2 illustrates the requirements in two interrelated groups A) Data collection, and B) Analysis of collected travel data.

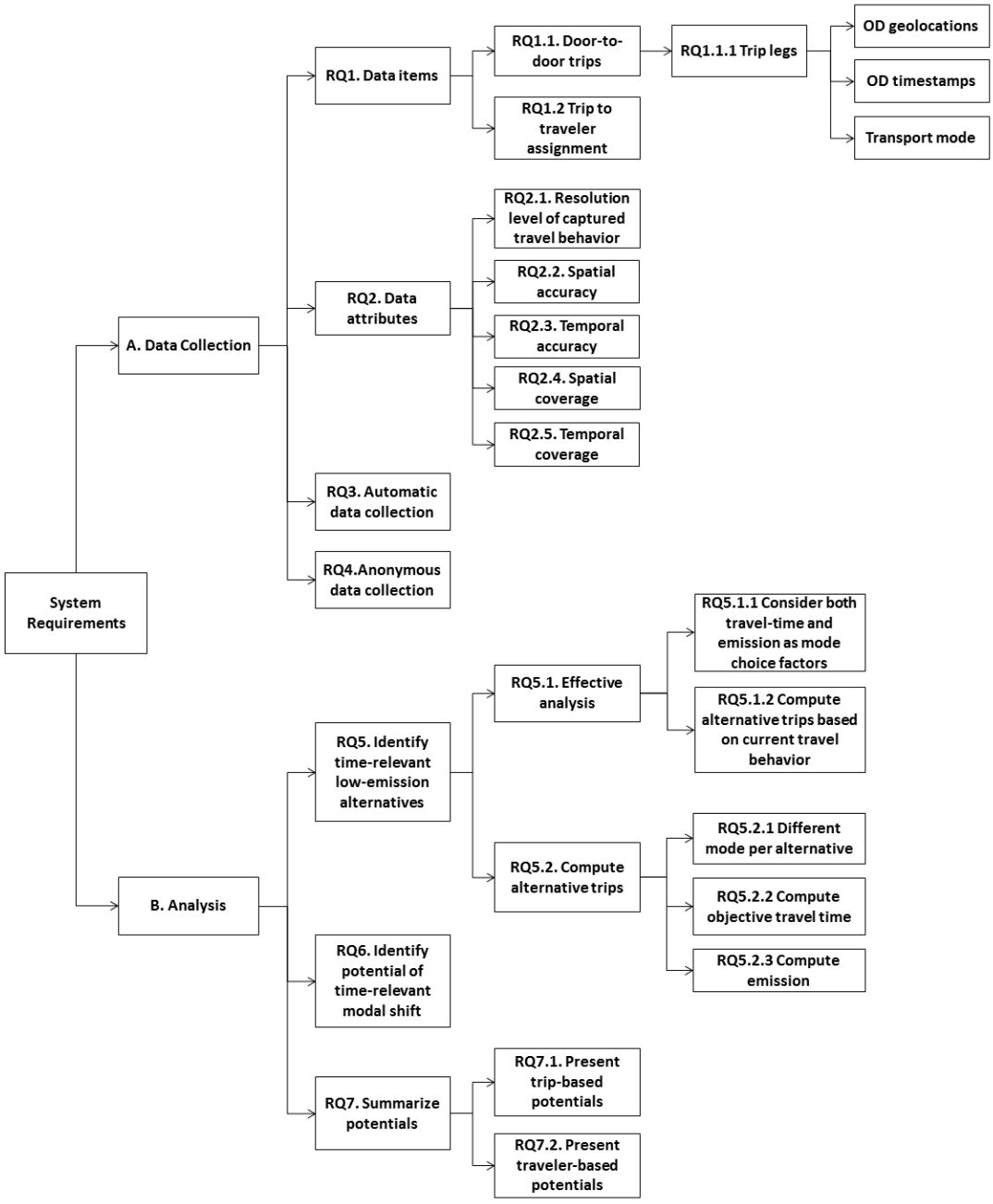

Figure 2: System requirements.

Software architecture and implementation:

We build our software system on top of the system initially implemented for the TrafficSense (TS) project [2], [3]. TS open-source software is explained in [5] and its source code, documentation and setup instructions are available on github [6]. TS system provides a mobile app to automatically collect, store and analyze mobility data of travelers. On top of that, this paper specifically implements requirements RQ1.1, RQ1.2, RQ5, RQ6 and RQ7 in the software.

Volunteers install the client-side TS app their mobile phones and keep the TS app active during their daily trips to automatically collect anonymous travel data. TS app also provides a menu for revision and confirmation of automatically detected modes. The app collects real-time mobility data, referred to as point-data, from GPS, accelerometer and other phone sensors using Google API for Android, and sends it over the Internet to the TS web server to be stored in a centralized database. HTTP, REST and JSON are used for data transfer. The point-data is collected at specific time intervals and includes timestamped geolocation (longitude and latitude) of traveler at each sampling interval along the trip route as well as an initial estimate of transportation mode, that is, *in-vehicle, walking, cycling,* or *idle*.

TS backend server is implemented in Python and PostgreSQL. TS backend filters and transforms the stored point-data. In particular, backend processing refines data points and modes of transportation to extract and store individual legs of each trip (requirement RQ1.1.1). Here, a trip-leg is each distinct section of a complete multimodal trip where traveler has used only one transportation mode. Another task of TS backend, is to compare the initial transportation mode detections against real-time locations of public transportation vehicles [7] as well as static schedule and routes. This way the in-vehicle legs are refined to distinguish between PT and private car as well as to specify more detailed PT classification such as bus, tram or metro. Travelers can also see their trip legs together with the transportation modes on the app and revise the detected modes for each leg in case needed. In addition, this paper implements further post-processing and data analysis using Python and PostgreSQL, and final visualization in Matlab. Details are explained in the following sections

Figure 3 illustrates the whole system architecture including components of both the original system as well as our implementations for this paper highlighted in bold.

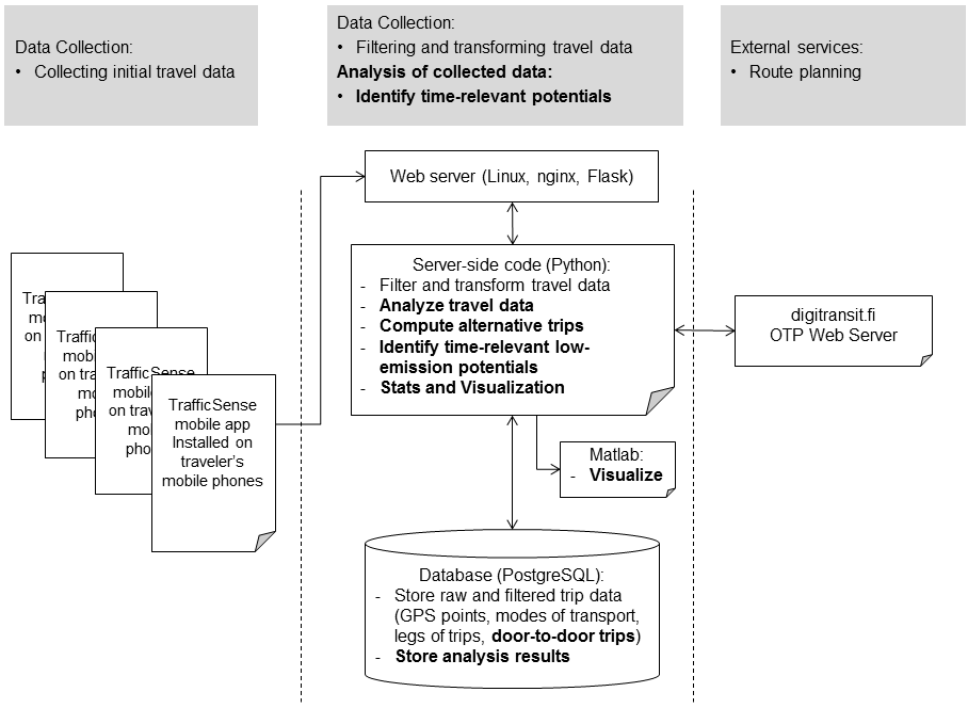

Figure 3: Components of the whole system including both the original TrafficSense system as well as our time-relevant analysis method. Implemented contributions of this paper are highlighted in **bold**.

Figure 4 is a screenshot of the client-side TS app installed on a traveler's mobile device.

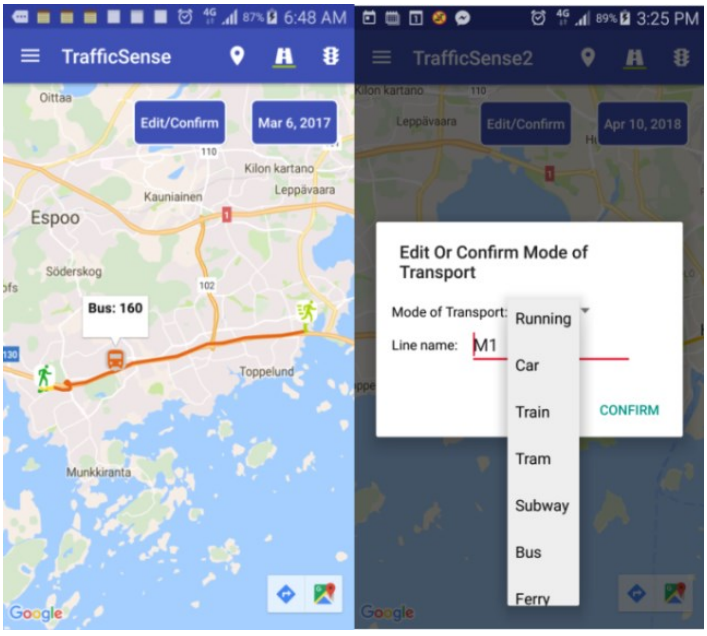

Figure 4: Screenshots of TrafficSense mobile app, where traveler's route and mode of transportation is shown (map data copyright of Google). If needed, traveler can also click on each trip-leg to revise the automatically detected modes.

An extracted door-to-door trip is shown in Figure 5.

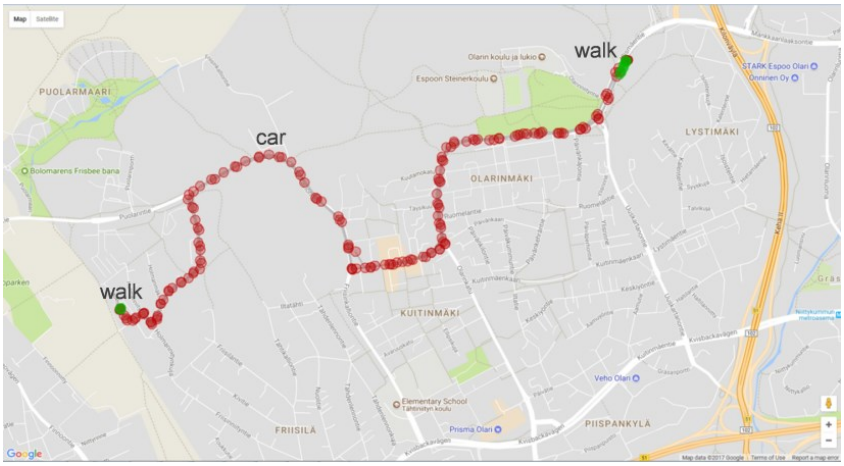

Figure 5: A door-to-door trip in Espoo - Finland, extracted by TrafficSense software from the collected travel data, and illustrated using Python and Google Maps API.

Link to source-code:

TS open-source software is explained in [5] and its source code, documentation and setup instructions are available on github [6] at https://github.com/aalto-trafficsense.

References

[1] K. Pohl, *Requirements Engineering : Fundamentals, Principles, and Techniques*. 2010.

[2] "TrafficSense - Research project at Aalto University," 2018. [Online]. Available: https://aaltotrafficsense.wordpress.com/about/.

[3] M. Heiskala, J. P. Jokinen, and M. Tinnilä, "Crowdsensing-based transportation services - An analysis from business model and sustainability viewpoints," *Research in Transportation Business and Management*, vol. 18, pp. 38–48, 2016.

[4] M. Salonen and T. Toivonen, "Modelling travel time in urban networks : comparable measures for private car and public transport," *Journal of Transport Geography*, vol. 31, pp. 143–153, 2013.

[5] M. Rinne, M. Bagheri, T. Tolvanen, and J. Hollmén, "Automatic recognition of public transport trips from mobile device sensor data and transport infrastructure information," in *International Workshop on Personal Analytics and Privacy (PAP 2017)*, 2017, vol. 10708 LNCS, pp. 76–97.

[6] "TrafficSenses software and mobile app - source code, documentation and setup instructions." [Online]. Available: https://github.com/aalto-trafficsense/.

[7] "TrafficSense public transport dataset." [Online]. Available: https://github.com/aalto-trafficsense/public-transport-dataset.