Towards Building Real-Time, Convenient Route Recommendation System for Public Transit

Garvita Bajaj, Rachit Agarwal, Georgios Bouloukakis, Pushpendra Singh, Nikolaos Georgantas, Valerie Issarny

Towards Building Real-Time, Convenient Route Recommendation System for Public Transit

Garvita Bajaj*†, Rachit Agarwal†, Georgios Bouloukakis†, Pushpendra Singh*, Nikolaos Georgantas†, Valérie Issarny†
*IIIT-Delhi, India, and †Inria-Paris, France
Email: *{garvitab, psingh}@iiitd.ac.in,
†{garvita.bajaj, rachit.agarwal, georgios.bouloukakis, nikolaos.georgantas, valerie.issarny}@inria.fr

***Abstract*—Public transportation is essential for sustainable and economical development of cities. Several transport organizations aim to provide service information to commuters through web and mobile *apps*. This information includes possible routes between two stations, estimated travel and arrival times, and real-time updates about traffic conditions. However, this information is currently not personalized according to commuter preferences. In this work, we emphasize the need for personalized transit service information to commuters and present a vision of our work in this direction. Our final goal is to develop a fully-functional personalized route recommendation system for public transit commuters. This involves identifying commuter preferences and suitable recommendation techniques, and developing a platform to communicate this information to the commuters. We identify the requirements for the development of this platform, and propose an architecture for our system. As a proof of concept, we present an Android participatory sensing application - *MetroCognition*, which acquires feedback on *convenience* experienced by commuters in public transit.**

***Index Terms*—ITS, Participatory Sensing, User Feedback, Recommendation System, Convenience, Queuing Networks**

I. Introduction

With rising urban population[1], there is a growing need for an efficient public transport system to make cities environmentally sustainable and economically competitive. Delhi, for example, introduced its metro network in 2002, owing to the increasing demands for better transit. Over time, the daily average ridership of the Delhi metro has increased from 80,000 in 2002 to 2.6M in 2015[2]. This number is expected to only increase with rising population. With the increase in demand for public transit, the comfort experienced by passengers in public transit decreases because of over-crowdedness. This makes it challenging for transit organizations to encourage people towards continuous use of public transit [1].

As an initial step towards solving this challenge and improving transport services, transport agencies and other organizations now provide mobile and web applications[3,4]. This leads to implementing the following improved services: *(i)* providing commuters with relevant and accurate information like schedules of transit, *(ii)* on-demand navigation support, and *(iii)* real-time traffic updates. To further improve the use of public transport, authors in [2]–[5] aim to personalize public transport services based on commuters preferences. Existing work to personalizing transport services can be broadly categorized into three approaches: *(i)* developing adaptive interfaces based on commuter context and historical data [3], *(ii)* developing algorithms for route recommendations based on commuter interests [2], [5], and *(iii)* passively identifying commuter preferences based on information stored in Automated Fare Collection Cards [4]. However, none of these existing approaches personalize commuter experience based on their *convenience* requirements. Recent works [1], [6]–[10] have highlighted the need to identify user convenience to improve public transit services. However, most of these works attempt to define "convenience" are objective in nature, and based on factors like time and crowdedness, for example, adjusting travel times to reflect convenience [8].

A subjective definition of user preferences can be found in [10], where authors have proposed an empirical model to identify "*hidden*" user preferences for route routing in multi-modal public transit. Their approach uses Stated Choice Experiments [11] to identify user preferences, but does not distinctively define the user parameters considered. Further, the use of Bayesian incremental learning makes their system computationally intensive, and may result in delays in route-planning. To overcome these challenges, we have defined convenience in public transit based on three parameters [7] - time, crowdedness, and seat availability. In this paper, we extend this work to propose a system – *Sarathi*[5], that identifies users' perception of convenience based on these parameters and uses it to personalize their public transit experiences.

Our approach to personalize a commuter's transit experience is currently limited to recommending the "*best*" path among the possible paths offered by multiple transit modes and their inter-connections. Identifying this personal "*best*" path for a commuter requires an understanding of his/her perception of convenience based on the parameters identified by the application. Further, since the transit network is highly dynamic (accidents/mishaps/traffic jams etc.), the recommendations must be provided to the commuters in real-time. Owing to intermittent and varying network connectivity within public transport [7], we identify the need for a middleware to

[1] ti.me/1dpszfx
[2] http://bit.ly/21bAEMt
[3] http://apple.co/1JNivAl
[4] http://bit.ly/1yXbumL

[5] "Sarathi" is a Hindi word which means "chariot"

facilitate the communication of these recommendations to the commuters in real-time.

In the following sections, we specify the requirements for this personalized route recommendation system (Section II). We present our system architecture (Section III). We present the initial choice of tools used to develop our system followed by the implementation details (Section IV). Our system is based on our previous work [7] where we have defined convenience based on three parameters - *i)* seat availability, *ii)* time delay, and *iii)* comfort experienced. Section V finally concludes our work.

II. System Requirements

In order to realize a personalized route recommendations system, we aim to have a client-server architecture (details in Section III). In order to reduce the processing time for identification of the "*best*" possible path, we delegate this processing to the server after identifying the list of possible paths using a navigation API. This leaves clients to have only UI requirements in order to allow commuters to interact with the system. The client ***UI*** module (or component) should provide a mechanism to address the following requirements:

1) ***Identifying possible paths***: In order to identify the "*best*" path, the client application should first identify all possible paths between a requested pair of stations. This can be done using a suitable *navigation* service. Some well-known navigation services include OpenStreetMap[6] and Google Maps Directions API[7]. Note that, other navigation services are also available, but due to limited space we are not listing them here.
2) ***Requesting Recommendations***: Once the navigation service provides a list of possible paths, the client must request the recommended convenience ratings (or feedback) for the listed paths, from the server. The server should identify the "best" path based on the feedback provided by the user (refer point 4), reorder the list of paths based on this feedback, and send it to the user.
3) ***Displaying Recommendations***: Once the server generates the re-ordered list of path recommendations, it should be conveyed to the client through a simple-to-understand and easy-to-use interface.
4) ***Requesting User Feedback***: Since user preferences change with time [12], it is important to continuously monitor this change by requesting their feedback. For this, the client should include a simple interface to allow users to submit their feedback for the chosen path.

The server, on the other hand, must perform all the computations required to generate the recommendations. Hence, it must include the following components:

- ***Online Recommendation System (RS)***: This component should be able to **learn, provide, and store recommendations**. The server should generate recommendations based on user's historic preferences, and preferences of other commuters who are similar to the user [13]. Further, since user preferences change with time and transport networks are dynamic in nature, the recommendations must be generated online to support real-time training and predictions. The identification of parameters for online training, such as the number of feedback considered and the weights of historical feedback, should be based on the performance of the metric considered to evaluate the recommendations (e.g., accuracy, precision, recall). The generated recommendations must also be validated by observing following two factors - *(i)* whether the user selects the most recommended path, and *(ii)* whether the user feedback for the selected path matches the predicted convenience level. Such validation would ensure continuous training of the online RS.
- ***Knowledge-base*** should be able to address **storage, scalability, security, and privacy** of the collected user feedback. Online RS require historical and real-time feedback for generating on-the-fly recommendations. This feedback should include information about the path taken between two stations, the time of travel, and rating values provided by users to the parameters considered in the definition of convenience. Generating on-the-fly recommendations would require frequent access to this information, thus, it should be stored such that the data can be accessed quickly. This storage of the data also requires scalability and security issues to be addressed. As the stored feedback contains private information, other than security of the data, user-management and privacy issues should also be addressed. These are explained next.
- ***User management***: Since our system will be used by a large number of users (commuters of public transport), it is important to manage them properly by creating user accounts corresponding to each user and providing adequate security features. Such security can be achieved using *authentication* and *authorization* of the user and the client respectively. Such aspects can be achieved by providing tokens either via using OAuth 2.0 or via OpenAM technology.

Apart from the client and server specific requirements specified above, the following components should have instances on both client and server:

- ***Communication component***: Intermittent network connectivity in public transport [7] may result in message delays or drops. This is undesirable in our application as the transport network is dynamic in nature, and a message may be crucial (requiring zero drop rate), or no longer relevant after some time (after acceptable delay). Thus, it is important to handle interactions between the client and the server such that clients receive every valid message from the server, and the server ensures delivery of valid messages to the clients. Such transmission issues arising due to intermittent network connectivity can be handled using a *middleware* which should be distributed in nature. The middleware can improve the communication between

[6]http://openstreetmap.org/
[7]http://bit.ly/1YgsmSY

the system entities, and provide valid messages, by using the following timing parameters: *i)* timeout; and *ii)* lifetime (e.g., setting a time-to-live for each request or data record). Below, we present the possible interactions that our middleware should support:

- *Synchronous Client-Server Interaction (sync CS):* the client sends a route request to the server and gets the recommendations within a *timeout* period. This is a simple request-response interaction and happens when the commuter is online (has Internet connectivity).
- *Asynchronous Publish/Subscribe Interaction (async PS):* this happens when the client sends a request to the server and gets the recommendations within a *lifetime* period. In this case, when the client is offline, it does not receive the recommendations promptly. Thus, the server's reply must wait during the lifetime period in a publish/subscribe infrastructure until the client is online. When the lifetime period expires, the response is lost.
- *One-way Client/Server Interaction (oneway CS):* this happens when the client submits a path feedback. This interaction is not crucial in terms of delay, but the message must be sent to the server as it will be used to train the recommendation system.

Based on the above possible interaction scenarios, our middleware must switch between the *sync CS* and *async PS* interactions, depending on client's connectivity. Furthermore, appropriate timing parameters should be applied. To do so, we aim to design the above interactions using *Queueing Network Models (QNMs)* [14], [15]. QNMs offer a simple modeling environment to evaluate the performance of a system. In our system, to represent the clients' intermittent connectivity, we utilize a specific queue (which we define as *ON/OFF queuing center*). Finally, we aim to capture the client connectivity and utilize it as a realistic input for our queuing model. The outcome allows us to tune the system's delay and improve it by applying appropriate timing parameters (*timeout* and *lifetime* periods).

- ***Privacy and security management***: The messages sent to/from the clients consists of personalized, and private information. This raises security concerns and requires our system to manage issues related to privacy and security of users at *Communication* (sending encrypted data), *Knowledge-base* (storing knowledge-base securely) and *User Management* component level.

Apart from the specific components described above, our system should also provide a feature to ***support extensibility***. Our previous work [7] defines convenience based on three parameters - seat availability, time delay, and comfort experienced. However, the concept of "convenience" is subjective in nature, and several other parameters, such as number of modal changes required in the path, walking time, and noise levels, can be used to define user convenience. Hence, our system must be generic in nature and should support an extensible representation of "convenience".

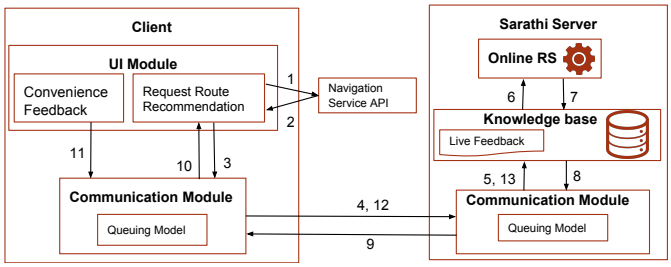


Fig. 1: Architecture of the *Sarathi* system with the different components and the flow of interactions between them.

III. System Architecture

Based on our requirements, we now propose an architecture for our system (see Figure 1). As explained above, we prefer to use a client-server architecture for our system to reduce the computation load on the client mobile device. Our focus is to shift most of the computation to the server by providing only an interface on the client with minimal load to interact with the system. The interface provided within the client application must be able to serve two purposes: *i)* sending feedback on commuter's convenience in a journey to the server, and *ii)* generating route recommendations for possible paths based on inputs from the server. The challenge associated with this real-time client-server interaction is the intermittent connectivity associated with mobile networks. We overcome this challenge by using a middleware to support multiple interaction paradigms to achieve higher performance and lower response times. The details on our middleware can be found in Section IV.

To identify the possible paths between a pair of stations, we use a *navigation* service which requires making a HTTPS call and returns the list of paths. Again, the use of navigation service is inspired by the goal of offloading computation from the client. This interaction between the UI and the navigation service is shown using arrows 1 and 2 in the figure. Once the list of paths is available, the client requests for the 'best' suited path as per user preferences. This requires sending the retrieved paths to the server. This communication between the client and the server happens via a middleware which uses a queuing model to ensure whether the messages sent from/to the client are still valid (arrows 3 - 5 in the figure). Once the message reaches the server, the server checks its knowledge-base to see if recommendations for the requested path exist. Based on the existing ratings, the online RS predicts the ratings of the paths requested by the client (arrow 6 in the figure). The choice of RS used and the format of ratings is discussed in detail in Section IV.

After the path ratings are predicted, the server stores them in the knowledge-base for further validation, and sends them back to the client using the middleware implementing the queuing model. This interaction is shown using arrows 7 - 9 in the figure. The client then receives the predicted

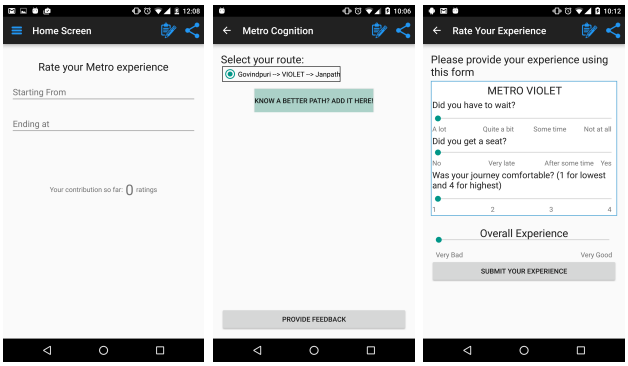
recommendations and its UI displays the re-ordered list of paths to the users (arrow 10 of the figure). On choosing a path, the client UI also provides an option to submit feedback to the server. This is done using a form on the client which sends the feedback to the knowledge-base at the server (arrows 11 to 13 in the figure).

IV. Implementation Details

Once the system requirements have been clearly defined, we identify the tools and technologies best suited to meet our goals. For the client application, we choose Android platform since it is the most popular smartphone OS in the market[8], and hence, has the largest user-base in the smartphone market. We develop an Android application – *MetroCognition* to allow users to request public transit routes between two stations. This app is currently limited to serve metro transits in Delhi (India), and Paris (France). However, it can be extended to other transit modes and cities as well. The UI component within *MetroCognition* is shown in Figure 2. A user enters the starting and ending stations (Figure 2a). Currently, we use Google Maps Directions API[7] to identify the possible transit routes between these stations (Figure 2b). A user can also add new paths (not listed by Google Maps Directions API) between these stations, which can later be stored to the knowledge-base and retrieved by other users. Thus, we rely on crowdsourced path information to provide more path options to users.

Once the possible transit paths are available, this information must be provided to the server to recommend the "*best*" path based on the user's convenience. This path information should be communicated in real-time as transport networks are dynamic in nature. However, the intermittent network connectivity with mobile networks makes it challenging to transmit information in real-time.

To improve the performance of our system (i.e., transmit/receive the information promptly), we intend to design and implement the *Sarathi middleware*. Particularly, to support heterogeneous interactions (*sync CS*, *async PS* and *oneway CS*), we plan to use our ongoing work on *eVolution Service Bus* (VSB). *VSB* is a framework for developing applications by leveraging existing middleware protocols, while handling interconnection when necessary. *VSB* is implemented by relying on our previous experience on XSB [16]. *Sync* and *oneway CS* interactions will be supported by using the CoAP[9] protocol, which is commonly used for IoT interactions. *Async PS* interactions will be supported by using the RabbitMQ[10] protocol (commonly used for mobile applications). Subsequently, our previous work on the analysis of timing constraints in heterogeneous middleware interactions [17] will enable our middleware to deal with intermittent network connectivity, and distinguish among the supported interactions to improve the *message delays* vs. *delivery success rates*. Moreover, estimated



(a) Station names (b) Recommended Paths (c) Feedback form

Fig. 2: Screenshots of *MetroCognition*

message delays derived using our queueing analysis will enable our middleware to apply appropriate timing parameters.

Currently, we rely on `GoFlow`[11] middleware to assist with client-server communication. The choice of `GoFlow` is motivated by the following: *(i)* it uses pub-sub messaging interaction paradigm which supports *async* interactions identified in Section II, *(ii)* it provides user management services at a middleware level, *(iii)* it ensures secure and private communication between clients and server, and *(iv)* provides a mechanism to store the knowledge gathered via MongoDB[12]. `GoFlow` queues all messages and sends them to/from client and server when Internet connectivity is present between the client and the server.

Our choice for the components running on the server is motivated by the difference in user preferences observed in different cities [7]. The differences in user preferences and the need for personalization requires a recommendation system. Since we rely on similarity in human preferences, user-based collaborative recommendations [18] and probabilistic matrix factorization [19] seem to be good choices. Also, authors in [20] highlight that including time-based information improves the performance of the recommendation engine, hence, we can also consider tensor factorization techniques [20] (BPTF) to predict convenience ratings for the users. We aim to compare these recommendation techniques for our dataset in our future work. Currently, the retrieved list of paths is personalized using weighed moving average of the historical ratings (on the client itself, with no contribution from the server). Once we identify the suitable recommendation technique for our application, we will integrate it with the *Sarathi* system.

Based on the recommendations generated by the online recommendation engine, the user can select a desired path from the recommendations retrieved for the list of available paths. The user may also provide feedback for the selected path using the form shown in Figure 2c.

The retrieved list is then personalized using weighted mov-

[8] http://bit.ly/1nSjC4A
[9] http://coap.technology/
[10] https://www.rabbitmq.com/
[11] http://goflow.ambientic.mobi/
[12] https://www.mongodb.org/

ing average of the historical ratings (on the client itself). We are working on the development of an online recommendation engine at the server to improve the prediction. Once the ratings are available, the user selects a desired path from the list of retrieved paths, and may provide feedback for the selected path using the form shown in Figure 2c.

Using *MetroCognition*, we have been also able to collect feedback from 9 users in Delhi, and 11 users in Paris. This data will be used to identify the best parameter values for training the recommendation system, and developing the queuing model for valid and timely message delivery. Such usage ensures realistic parameterization of the system. Using the proof of concept implementation, we are able to justify the vision we have for building a system that personalizes the transit based on the convinience.

V. Conclusion

In this paper, we identify the need for personalized route recommendations in public transit based on commuters' perception of convenience. Our goal is to develop a system which can infer commuter preferences of convenience in public transit, and provide suitable route recommendations. We propose our system architecture and present the requirements of the different components identified in the architecture. We also present the initial choice of tools and technologies to realize this system. In the future, we aim to extend our middleware to other transit based applications to support interactions between the client and the server.

VI. Acknowledgment

The authors would like to thank the collaborative research associate team "SARATHI-Personalized Mobility Service for Urban Travelers" for this research. This research has been conducted within the context of the joint targeted program in Information and Communication Science and Technology-ICST, supported by CNRS, Inria, and DST. This work is also partially supported by: *i*) the European project "Federated Interoperable Semantic IoT/cloud Testbeds and Applications (FIESTA-IoT)[13]" from the European Union's Horizon 2020 Programme with the Grant Agreement No. CNECT-ICT-643943, and *ii*) ITRA project, funded by DEITy, Government of India, under grant with Ref. No. ITRA/15(57)/Mobile/HumanSense/01.

References

[1] Y. Yim and A. Ceder, "Smart feeder/shuttle bus service: consumer research and design," *Journal of Public Transportation*, vol. 9, no. 1, p. 5, 2006.

[2] B. Ludwig, B. Zenker, and J. Schrader, "Recommendation of personalized routes with public transport connections," in *Intelligent interactive assistance and mobile multimedia computing*, pp. 97–107, Springer, 2009.

[3] H. Nakamura, Y. Gao, H. Gao, H. Zhang, A. Kiyohiro, and T. Mine, "Toward personalized public transportation recommendation system with adaptive user interface," in *Advanced Applied Informatics (IIAIAAI)*, pp. 103–108, IEEE, 2014.

[4] N. Lathia, J. Froehlich, and L. Capra, "Mining public transport usage for personalised intelligent transport systems," in *2010 IEEE International Conference on Data Mining*, pp. 887–892, IEEE, 2010.

[5] G. Tumas and F. Ricci, "Personalized mobile city transport advisory system," *Information and Communication Technologies in Tourism 2009*, pp. 173–183, 2009.

[6] M. Wardman, "Valuing convenience in public transport," p. 73, 2014.

[7] G. Bajaj, G. Bouloukakis, A. Pathak, P. Singh, N. Georgantas, and V. Issarny, "Toward enabling convenient urban transit through mobile crowdsensing," in *IEEE ITSC*, pp. 290–295, IEEE, 2015.

[8] T. Litman, "Valuing transit service quality improvements," *Journal of Public Transportation*, vol. 11, no. 2, 2015.

[9] R. Cayford and Y. Yim, "Personalized demand-responsive transit service," 2004.

[10] J. Zhang and T. A. Arentze, "Personalized multi-modal route planning: a preference-measurement and learning-based approach," in *MOBIQUITOUS*, pp. 338–344, ICST, 2014.

[11] J. J. Louviere, D. A. Hensher, and J. D. Swait, *Stated choice methods: analysis and applications*. Cambridge University Press, 2000.

[12] Y. Koren, R. Bell, C. Volinsky, *et al.*, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[13] G. Shani and A. Gunawardana, "Evaluating recommendation systems," in *Recommender systems handbook*, pp. 257–297, Springer, 2011.

[14] E. D. Lazowska, J. Zahorjan, G. S. Graham, and K. C. Sevcik, *Quantitative system performance: computer system analysis using queueing network models*. Prentice-Hall, Inc., 1984.

[15] F. Baskett, K. M. Chandy, R. R. Muntz, and F. G. Palacios, "Open, closed, and mixed networks of queues with different classes of customers," *Journal of the ACM (JACM)*, vol. 22, no. 2, pp. 248–260, 1975.

[16] N. Georgantas, G. Bouloukakis, S. Beauche, and V. Issarny, "Service-oriented Distributed Applications in the Future Internet: The Case for Interaction Paradigm Interoperability," in *ESOCC 2013 - European Conference on Service-Oriented and Cloud Computing*, (Malaga, Spain), Sept. 2013.

[17] A. Kattepur, N. Georgantas, G. Bouloukakis, and V. Issarny, "Analysis of Timing Constraints in Heterogeneous Middleware Interactions," in *ICSOC'15 - International Conference on Service Oriented Computing*, (Goa, India), Nov. 2015.

[18] R. Burke, "Hybrid recommender systems: Survey and experiments," *User modeling and user-adapted interaction*, vol. 12, no. 4, pp. 331–370, 2002.

[19] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," in *NIPS*, vol. 20, pp. 1–8, 2011.

[20] H. Pragarauskas and O. Gross, "Temporal collaborative filtering with bayesian probabilistic tensor factorization," 2010.

[13] http://www.fiesta-iot.eu/