OptiPath: Optimal Route Selection Based on Location Data Collected from Smartphones

C. Kalampokis, D. Kalyvas, I. Latifis, and V.A. Siris

Mobile Multimedia Laboratory
Department of Informatics
Athens University of Economics and Business
vsiris@aueb.gr

Abstract. We present a system that selects the optimal route between two points for vehicles moving in an urban environment. Our approach uses location data gathered from smartphones, which includes route segments and their corresponding travel time. The system can be used as a standalone application with a user interface for path visualization. Additionally, route selection and location prediction can be used for improving streaming video applications and for scheduling delay tolerant data transfers to achieve mobile data offloading.

1. Introduction

Vehicular traffic congestion is a problem that concerns the inhabitants of urban areas. The problem of selecting travel paths that minimize the total travel time, thus accounting for congestion, has been investigated in the past, e.g. [1], while prior work has also investigated the use of wireless vehicular networks to gather information to select the best (shortest) route [2]. In this paper we present a client-server system (Fig. 1) for collecting real-time travel data over a cellular network from smartphones with geo-position sensors. This real-time travel data is used together with the list of possible routes from Google Maps, to select a route between an origin and a destination with the smallest travel time. The system consists of an Android client application called "OptiPath" and a multithreaded Java-based server with a database that contains the collected travel data.

The client, while in passive mode, records the route that is covered accompanied by real-time traffic information, such as speed and duration. The route consists of smaller segments that will be referred to as route tokens. Each route token contains the GPS coordinates of the route segment's start and end points, the speed, and the duration. Periodically, the route tokens are sent to the server using an Android service. The client's operation in passive mode operates as a seamless background process. The scalability of this approach is similar to that of other crowd-sourcing applications, e.g., to build a database with GPS and network bandwidth information [3].

In active mode, the user can use the client to select a desired destination, which together with his current location are placed in a query that is sent to the server, requesting the route between the two end-points with the shortest travel time. When the server receives a route query, it in turn sends Google Maps queries using the

Google Directions API[1], in order to obtain alternative routes between the two designated points. After it receives the alternative routes, the server computes the travel time for the alternative routes utilizing the relevant data stored in the database, and returns the route with the shortest travel time back to the OptiPath client. At the client, the server's response is graphically presented on a map.

2. System Operation

As we have outlined above, our architecture consists of two modes of operation, passive and active. Next we present in more detail the operation of each mode.

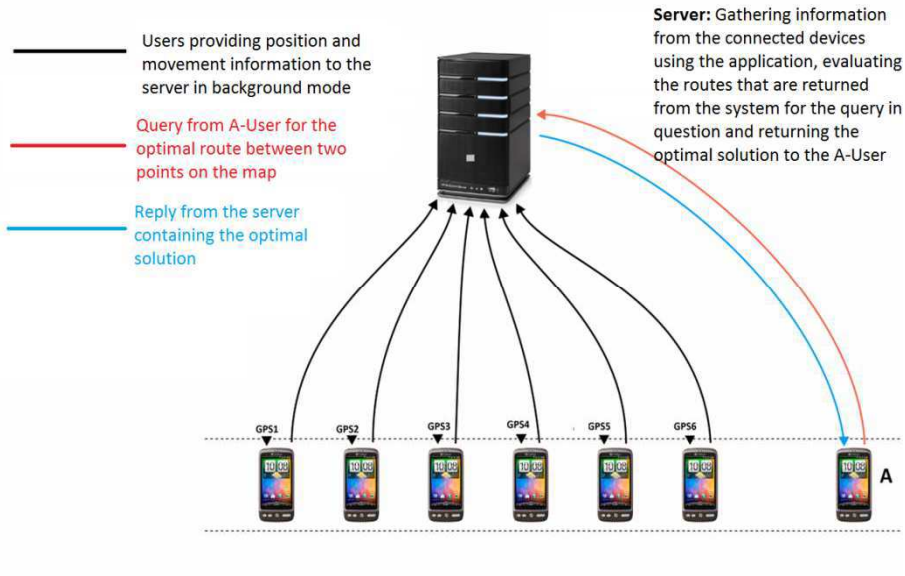

Fig. 1. Architecture and operation of OptiPath

Passive Mode:

On the client's side, OptiPath regularly creates route tokens by obtaining the GPS coordinates from the smartphone's sensor. After several experiments, we have concluded that the best accuracy is achieved when route tokens are created every ten meters. Such an interval provides sufficient accuracy when the vehicle turns at crossroads. Nevertheless, the interval's duration reflects a tradeoff between accuracy on one hand, and 3G bandwidth and mobile device processing power on the other hand. Creating tokens at fixed periods of time was rejected in order to avoid accepting zero-length tokens when the user remained stationary, e.g., during a traffic jam or

[1] Google Directions API Specification:
https://developers.google.com/maps/documentation/directions/

when the application is left running when not in use. Once several samples are collected, the application creates a list of tokens and sends it to the server, using a background Android Service. At the server's side, the route tokens received pass an integrity check and are inserted into the database.

Active Mode:

The user can choose a destination by touching a point on OptiPath's map or by searching an address using either the Google Places API[2] or the Google Places Autocomplete API[3]. The application creates a token that consists of the current location and the selected destination coordinates, which is forwarded to the server. When the server receives the token, it uses the coordinates of the starting and ending point, and creates a query to Google Maps for alternative routes between these two points. Google responds by sending one or more routes, consisting of tokens with different lengths compared to the tokens stored in database. Afterwards, the server creates a list of tokens for each alternative route. These lists are evaluated one by one using tokens stored in database according to the following procedure: Initially, the tokens from each route are mapped to possible matches contained in the database. The database tokens that are selected and eventually contribute to travel time estimation are those that overlap with the tokens from the proposed route either fully or partially. Note that the database tokens are usually shorter, since they are created every ten meters travelled distance. The matching procedure accounts for the possible lack of precision of the GPS signal by considering relaxed boundaries to determine the matching tokens. Specifically, the matching procedure involves checking if the database tokens are contained either fully or partially in the parallelogram that is created based on the Google route token with the corresponding error margins.

More recent tokens have a higher influence on the travel time estimates. Specifically, we group token samples based on when they were added into the database in three sets: route tokens added in the last 10 minutes, route tokens added 10-20 minutes ago, and route tokens added 20-30 minutes ago, which contribute to the travel time estimation with weight 60%, 25%, and 15%, respectively. We do not consider samples older than 30 minutes. If a token of the proposed Google route does not have a match in the database, then the precomputed Google estimation is considered in the travel time estimation.

After the above procedure, the optimal route with the shortest travel time, among the alternate routes provided by Google Maps, is returned to the client in a message that in addition to the estimated travel time also contains the total distance, the list of tokens, and the turn-by-turn directions, as provided by the Google Maps API. If the

---

[2] Google Places API Specification:
http://code.google.com/apis/maps/documentation/places/

[3] Google Places Autocomplete API Specification:
http://code.google.com/apis/maps/documentation/places/autocomplete.html

user deviates from the proposed route for some amount of time, then the application recommences the procedure, querying the server again for a new route and recalculating the optimal route from the current point to the initial destination, which it finally returns to the client.

3. Real world execution

To validate OptiPath we used two mobile devices with 3G connectivity, while the server was hosted on a remote computer. Two vehicles travelled along different routes. The testing location is shown in Fig. 2, where the two alternative routes A and B from the same start and end point are shown with different colours. The collection of tokens starts at Log point, hence Google Map's precomputed estimate is used as the travel time for the segment from the starting point to the log point.

Fig. 2. Two alternative routes between start and end points

We carried out two tests where the vehicles travelled from start to the end point several times. In the first test, vehicle 1 moved along route A (Fig. 3, left) with a higher average speed than vehicle 2, which moved along route B (Fig. 3, right). After completing the collection of tokens, the client sent a query for the optimal route between the start and end points. As a response to the query, route A was returned with score 180, against route B that had score 250; the score corresponds to the estimated travel time in seconds. In the second test, vehicle 1 moved along route A with a lower average speed than vehicle 2, which as before moved along route B. After completing the collection of tokens, the client created a query for the optimal route between starting and ending point. According to the response, route B was now evaluated with score 190, whereas route A received score 200.

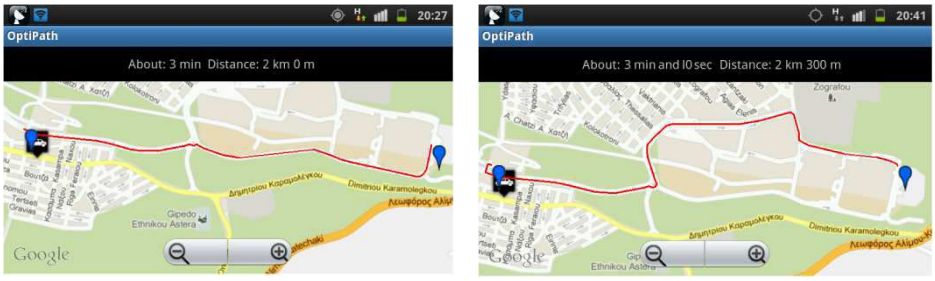

Fig. 3. Screen shots from the OptiPath client that depict the two alternate routes

4. Conclusion and Ongoing Work

We have presented the implementation of a system for determining the optimal route between two end-points based on the shortest travel time. The system utilizes data that includes route segments and their corresponding travel times, which are obtained from smartphones using crowd-sourcing.

Aside serving as a stand-alone application, the system's route selection and location prediction can be utilized for improving the performance of video streaming [3] and for scheduling delay tolerant data transfers to offload mobile traffic from cellular networks to WiFi hotspots [4]. These are directions we are currently investigating. In particular, the presented system can be enhanced or combined with a database containing the achievable throughput of a cellular network in different locations. This information combined with the prediction of the route and the position in different time instants allows planning of the transfer rate to avoid buffer under-runs, thus improving the performance of video streaming, while minimizing the maximum amount of buffer required [3]. Another direction involves the transmission of large files such as movies and pictures which are delay tolerant. This is becoming particularly popular with the increasing use of smartphones and mobile social networks, where users upload short clips or pictures in a delay tolerant manner. For such an application, estimates of both the cellular and the WiFi hotspot throughput in different locations are necessary. The goal would be to utilize as much as possible WiFi hotspot access along the selected route, thus minimizing the use of cellular networks, while satisfying a maximum delay threshold [4]. Another interesting direction is to utilize route and location prediction to perform pre-fetching or proactive caching [5,6], which can be used to increase the performance of video delivery in cellular networks [7], and reduce the peak load of mobile networks by offloading traffic to WiFi hotspots [8].

References

[1] James F. Campbell, "Selecting routes to minimize urban travel time," Transportation Research Part B: Methodological, Volume 26, Issue 4, August 1992, Pages 261–274.

[2] Kevin Collins and Gabriel-Miro Muntean, "An adaptive vehicle route management solution enabled by wireless vehicular networks," in Proc. of IEEE VTC 2008-Fall.

[3] H. Riiser, P. Vigmostad, C. Griwodz, and P. Halvorsen, "Bitrate and video quality planning for mobile streaming scenarios using a GPS-based bandwidth lookup service," in Proc. of ICME, 2011.

[4] A. Balasubramanian, R. Mahajan, and A. Venkataramani, "Augmenting mobile 3G using WiFi," in Proc. of ACM MobiSys Conference, 2010.

[5] P. Deshpande, A. Kashyap, C. Sung, and S.R. Das, "Predictive Methods for Improved Vehicular WiFi Access," in Proc. of ACM MobiSys 2009.

[6] V.A. Siris, X. Vasilakos, and G.C. Polyzos, "A Selective Neighbor Caching Approach for Supporting Mobility in Publish/Subscribe Networks," in Proc. of ERCIM Workshop on eMobility. Held in conjunction with WWIC 2011.

[7] N. Golrezaei et al, "FemtoCaching: Wireless Video Content Delivery through Distributed Caching Helpers," in Proc. of IEEE Infocom 2012.

[8] F. Malandrino et al, "Proactive Seeding for Information Cascadesin Cellular Networks," in Proc. of IEEE Infocom 2012.

DYMO Routing Protocol with Knowledge of Nodes' Position

Enrica Zola, Francisco Barcelo-Arroyo

Universitat Politècnica de Catalunya, c/ Jordi Girona 1-3, Mòdul C3
08034 Barcelona, Spain
{enrica, barcelo}@entel.upc.edu

Abstract. Knowledge of the physical location of the nodes can improve the efficiency of routing protocols in Mobile Ad-hoc Networks (MANETs). In the Beacon-Less Routing (BLR) algorithm, for instance, the source node broadcasts its data packets while the forwarding node is selected in a distributed manner among all its neighbouring nodes. We propose to apply the forwarding strategy of BLR to the route discovery process of Dynamic MANET On-demand (DYMO) routing protocol. Assuming that position information is available at transmitting nodes, route request (RREQ) packets will be broadcasted with such information. All receiving nodes will compute whether they are in the forwarding area and, if so, they will calculate a delay which will be applied to the next RREQ broadcast. The node with lower delay will resend the RREQ first, thus selecting, in a distributed manner, the best located node in the forwarding area. With this modification in the DYMO protocol, it is expected that the amount of RREQs circulating in the network will reduce, thus lessening the routing overhead and improving the overall throughput.

Keywords: MANET; routing protocol; DYMO; DFD; localization.

1 Introduction

The task of routing in Mobile Ad-hoc Networks (MANETs) has to be performed by the user nodes. These nodes are mobile and, sometimes, unreliable. Knowledge of the physical location of the nodes can improve the efficiency of routing. Many protocols have been proposed: some are intended to minimize the search space for route discovery towards the destination node [1]; others apply the source routing in order to establish the geographical path that a packet has to follow towards its destination [2]. Still, both approaches share the idea of flooding the route requests (RREQs) through the network in order to establish a path to destination before sending data to it. Other authors propose to flood data packets in the network, without the need for routing table set up and maintenance. In the Beacon-Less Routing Algorithm (BLR) [3], for instance, the source node broadcasts its data packets while the forwarding node is selected in a distributed manner among all its neighbouring nodes. The knowledge of nodes' positions is a requirement in the BLR approach. As a contention-based