easy-City: a route search system for public transport users[*]

Fábio Pereira and João Barreto
Instituto Superior Técnico, Technical University Lisbon / INESC-ID Lisbon
[fabio.pereira, joao.barreto]@ist.utl.pt

ABSTRACT

This paper addresses the hard problem of route searching on public transport transportation. easy-City is a decentralized collaborative system that takes advantage of public transport users' mobility and observations to provide a route searching service. Users' mobile phones and deployed infra-structure devices establish a delay-tolerant network to efficiently disseminate information about routes and other events occurring on the public transportation network. We introduce a novel message dissemination protocol that exploits estimates of each user's future locations to make informed forwarding, message aggregation and buffer management decisions.

A simulation-based evaluation of easy-city with geographical and public transport data shows that, in a realistic scenario, the system aids users to find best route options even in the presence of unpredictable events. A simulation with 468 nodes shows that easy-City often provides better options than a typical journey planning application based on static knowledge about the transportation network.

Keywords: public transport systems, delay-tolerant networks, ubiquitous computing, store-carry-forward

1. INTRODUCTION

As population densities in big urban centers grow, sustainable mobility is assuming more and more importance. Gradually, more public transport alternatives are made available to citizens. While this change substantially improves one's mobility, it also makes it more complex to take advantage of public transports efficiently. The central question that public transport users repeat routinely is "what is the best way to get from A to B by a combination of public transports?". Finding an adequate answer to the question in a reasonable amount of time is crucial to the perceived quality of the public transport service.

Given two points in a city, there are typically several ways to travel from one point to the other. Alternatives can vary in many ways, like means of transportation (e.g., subway, train, tram, bus, ferries), operators, estimated durations, prices, number of tickets needed to completion, etc. All those aspects are important for public transport users and they can have a massive impact on the effective outcome of their routing choices.

An ideal solution to this problem should fulfil two main requirements. First, it should provide the user with the best route option in terms of duration, price and number of times the user needs to change transport (or a possible conjugation of these factors, according to user requirements). Second, it should maximize the availability of the route search service, if possible, allowing its usage anytime, anywhere.

However, a number of fundamental challenges push current state-of-the-art systems away from the ideal solution. Information about the transportation network is typically numerous and complex. So, delegate the responsibility of finding the best option to the user is not a good choice.

Not surprisingly, public transport journey planners became very popular due to the automatic route computation performed by these systems. However, journey planners are often centralized systems, which can be accessed by an Internet connection, or off-line using locally cached information about the public transportation environment.

Furthermore, the information is often dynamic, it changes constantly and those applications often use static sources of information. Hence, real-time unpredictable events are not reflected on the decisions taken by these journey planners. To overcome this challenge, several public transport operators started to introduce sensors on the infrastructure, to gather real-time information about their services (waiting times, estimated durations, traffic jams, accidents, service interruptions). This information is typically disseminated through the infrastructure to be available where it is necessary. However, these solutions are very expensive to install and maintain [7]. So, the coverage of these solutions is usually limited (few sensors, few access points). Furthermore, the information gathered is typically related to one specific operator instead of all the public transport offer. So, these systems usually inform the user about the best route option concerning the services of the system owner.

This paper advocates that users should be involved in the route planning system, not only as interested consumers but also as collaborative feeders of dynamic information they gather on their journeys. As we show later on, leveraging an existing infrastructure-based system with the information provided by crowds of users can substantially enrich the ability of the system to propose adequate routes.

[*]This work was partially supported by FCT (INESC-ID multi-annual funding) through the PIDDAC Program funds

In this paper we introduce the easy-City middleware. To the best of our knowledge, easy-city is the first collaborative decentralized route planner for public transport users, based on an ad-hoc networking approach. In easy-city, commodity wireless devices carried by users work together with inexpensive wireless devices installed in public transport vehicles, stations and stops. Users gather information about real-time journey times. Stop nodes produce infra-structure warnings about the public transportation network, like accidents, traffic jams and service information. Vehicles exchange messages with users to inform them when they enter or exit them.

Using the gathered information and a transport network map, easy-City lets users ask for routes between two given points. Maps have information provided by operators about the public transport offer on a city, like travel times, waiting times and pedestrian times. Then, the knowledge on the map is complemented by the gathered information through time estimative calculations. The gathered information is divided in individual observations and they are organized on an observations base.

easy-City relies on a message dissemination protocol which uses estimates of each node's future locations to make forwarding, buffer management and message aggregation decisions.

Based on simulations with real world city maps, we show that users aided by easy-City frequently (20 to 49% times) finish their journeys substantially earlier than they would if they relied on using traditional static journey planners.

The remainder of this document is arranged as follows. Section 2 surveys related work. Section 3 describes easy-City and the underlying mechanisms. Section 4 then presents the results obtained by simulating easy-City in different real world scenarios. Finally, Section 4.2 draws conclusions.

2. RELATED WORK

In this section we describe the traditional approaches to solve the route planning in public transportation problem. Some of these approaches have been in use for years, others are more recent ones.

In the most primitive solution to the problem, the public transport user performs all the tasks of route planning in public transportation. Using static sources of information, the user behaves as an optimizer, performing cognitive tasks to find a route option to travel between two given points. However, this approach has clear limitations. The main issue is that, typically, humans do not perform well in finding the best route option between two given points [15]. This issue becomes more problematic when the complexity of available options increases.

An evolution on state-of-the-art solutions was brought by the introduction of infra-structure sensors by operators on public transportation networks (e.g. RATP geographical location system [20], CTA's BusTracker [2], the Tube electronic displays [18]). Typically, these sensors capture information about the environment. Then, captured information is made available on different information sources (e.g., electronic panels, SMS, email). By accessing these information sources, users have more and better information to base their decisions. However, in this approach the user still performs the main tasks on finding the best route option. Nevertheless, the error probability is still high which can drive to bad decision making.

With the growth of Internet popularity, the emergence of automatic Internet-based route planning in public transportation (e.g. Google Transit [10], London Journey Planner [17]) happened naturally. However, the need for an Internet connection is an important limitation. Since route optimizers are more useful when users are travelling or about to travel, the use of such systems comes often with a considerable cost, both in terms of monetary fees and battery consumption. Furthermore, these systems often do not use real-time information about the transportation network. Hence, they fail to reflect the effects of events like accidents, traffic jams and service interruptions on the route planning suggestions.

The field of Vehicular Ad-Hoc Networking (VANET) deals with problems such as road warning applications and navigation systems, which share important resemblance with the problem that we address in this paper. Many VANET solutions are based on delay-tolerant networking [6], possibly backed up by some road-side infra-structure [16]. The main challenge on delay-tolerant networking is the communication between nodes. Due to the lack of simultaneous reliable end-to-end communication paths between nodes, typically, a Store-Carry-Forward mechanism is used to deliver messages. The effectiveness of such systems depends strongly on adequate message forwarding and dissemination mechanisms [25, 8, 9, 24, 23, 14, 3, 22, 26], buffer management policies [4, 13] and data aggregation mechanisms [1, 16, 19].

However, there are some differences between those VANET solutions and our scenario. Users may migrate between vehicles during their route, rather than relying on the same vehicle for the whole journey. The mobile devices that users carry have significantly higher energy constraints than the devices deployed in vehicles. Users have less predictable routes, because their movement patterns are not limited to road infrastructures. Finally, most means of public transport obey to schedules, instead of purely continuous traffic as in automobile traffic.

3. EASY-CITY

In this section we start by describing the system model explaining the basic assumptions, rules and components necessary for the deployment of the easy-City system. We then introduce easy-City, starting by its basic features, and then detailing more advanced mechanisms of our solution.

3.1 System Model

There are three node types: users, vehicles and stops. User and vehicle nodes are mobile. They represent, respectively, public transport users and public transport vehicles. Stop nodes are a representation of a specific stop point on the transportation network (stations, docks, bus stops). Nodes are able to communicate in short-range opportunistic interactions through some wireless communication protocol, such as Bluetooth [21].

As user nodes move through the transport network, they approach stops and enter vehicles. When users intersect stop nodes, they receive *location-time messages*. Location-time messages have a location associated to a temporal instant. A location-time message reports that an user, that receives it, is standing on the reported location, on the report time instant.

When users intersect vehicle nodes, they receive *vehicle-time messages*. Vehicle-time messages have a vehicle identifier and a temporal instant. A vehicle-time message reports that an user enters, is standing at or leaves the vehicle identified by the contained identifier.

The main role of stop nodes is to introduce observations reporting consequences of unpredictable events that occur on the transportation network. For example, when a service interruption occurs, the waiting time for the service increases. Stop nodes affected by the interruption introduce observations reporting that the waiting time for that service increased.

3.2 Basic Solution

easy-City is a collaborative decentralized system. It establishes a delay-tolerant network between nodes. The goal of easy-City is not only to be a one-time searching application, but also a real-time advisor, reporting changes on the environment, including invalidations on the current best route option and fresher best options. As we explain next, the key insight of easy-City is that each user node voluntarily contributes to enrich easy-city's ability to provide accurate and up-to-date route suggestions.

Each user node maintains the following state:

- a map with information about the transportation network, like stop locations, vehicle-travelling times, walking times and estimated time of arrival;
- information about services on the transportation network, like transport schedules and routes;
- an observation base (explained next).

At any time, the user can access his node to request a route between two given points. The node uses the knowledge present on the map and the observations present on the observation base to compute an answer. Later, we explain how this task is performed.

A map is a weighted graph representing the transportation network, where nodes are points on a city, stations or stops, and edges are transitions between nodes which correspond to the action of moving from one node to another. There are three kinds of edges: waiting edges, vehicle edges and pedestrian edges. Waiting edges represent the action of waiting for a transport. Vehicle edges represent the action of travelling between two stops or stations on a vehicle. Pedestrian edges represent the action of moving between geographical points on the city by walking. The weight of an edge represents the time it takes to move from the source to the destination node of the edge.

3.2.1 *Observations*

An observation is a tuple that represents the time between two events, as observed by some user node. There are two kinds of observations: waiting time observations and travel time observations. A waiting time observation represents the time from the moment where the user arrived at a given stop and the instant where he got in the vehicle he was waiting for. Waiting times are constructed by combining the location-time message the user received from the stop and the first vehicle-time message the same user received once he entered the vehicle.

In contrast, travel time observations report travelling times between consecutive stops while a user is ridding a transport. Travel time observations are captured by measuring the delay between the reception of different location-time messages when the user is inside a vehicle.

All user nodes travelling through the transport network create observations about their journeys, reporting experienced travel and waiting times. Figure 1 illustrates a route option being travelled by an user node and the correspondent reported times through observation messages dissemination.

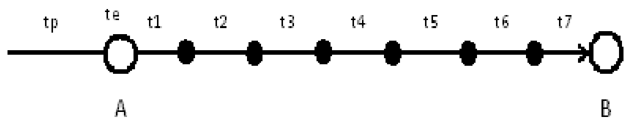


Figure 1: A-B Route and observed times

The waiting time in A (t_e) is given by the delay between the reception of the first location-time message from A and the vehicle-time message from the vehicle that user wants to take. The first travel time is given by the delay between the reception of the vehicle-time message and the reception of the next stop's location-time message. The following travel times are measured by the delay on the reception of different location-time messages.

User nodes disseminate created and received observations. We show later on, how user nodes perform dissemination.

3.2.2 *Route Planning*

When the user at a given node wants to find the best route option between two given points, its node uses the map together with the observation base and a graph search algorithm. Tuples on the observation base are used to update the times (weight of edges) on the map. This way, we enhance the base map with the flexibility to reflect possible changes on the environment. On the updated map, we apply Dijkstra's search algorithm [5] to find the best path.

For each waiting edge or travel edge on the local map of a user node, there is a correspondent tuple on the observation base. An observation base is a set of entries in the following form:

$$\begin{pmatrix} o_b \\ o_1 \\ o_2 \\ o_3 \\ \ldots \end{pmatrix}$$

where o_b represents a base observation, part of the node's initial knowledge about the transportation network. The remaining entries are observations created on-the-fly either by the user node itself or by some other user nodes.

When an user node receives an observation, it adds an entry on the correspondent register on the observation base.

The current estimate of a time on the map is calculated as a weighted average of all known observations for that time, as follows.

$$\bar{t} = \frac{\sum_{i=1}^{n} P_i t_i}{\sum_{i=1}^{n} P_i} \times (1 - \alpha) + t_b \times \alpha$$

Static times (t_b) have a weight of α on the weighted average. The α value is configurable. A high value represents a high weight for the static component on the calculation of estimated times. If there are no user observations, about a given segment on the map, the static component weight is 1.

The weight of user observations is not uniformly distributed. A recent observation is more reliable to represent the current state of an edge. Older observations must be carefully used, because they can become inaccurate.

The following function computes the weight of an observation:

$$P(i) = \begin{cases} t_{val \to i} - t_c & : t_{val \to i} > t_c \\ 0 & : t_{val \to i} \leq t_c \end{cases}$$

The weight of an observation is given by the difference between the expiration instant (t_{val}) and the current instant (t_c).

After the computation of the weighted average, the returned result is used on the searches for the best route option. To keep the user in track of public transport network changes, in this basic solution the best route option is calculated in each observation message reception, using the Dijkstra algorithm.

For example, to the following register:

$$\begin{pmatrix} 300 \\ 1294(tval = 1360) \\ 1365(tval = 1211) \\ 266(tval = 2900) \\ ... \end{pmatrix}$$

the current time ($t_c = 900$) estimate is given by:

$$\bar{t} = \frac{1294 \times 490 + 1365 \times 311 + 266 \times 2000}{490 + 311 + 2000} \times (1-\alpha) + 300 \times \alpha$$

This example may represent a waiting time edge of a stop that suffered some delay (1294, 1365) due to a service interruption and now returned to normal operation (266).

3.3 Advanced Mechanisms

When deployed on a realistic scenario, easy-City needs to deal with important resource constraints, where resources such as memory, CPU, bandwidth and battery limitations that are common to many mobile devices and networks. This section introduces the mechanisms that make easy-City appropriate for the scenarios that this paper addresses.

3.3.1 Message Dissemination

We introduce a dissemination mechanism that uses context information to take more efficient forwarding decisions.

Each node has a transport buffer to store observation messages. When a node meets another, our protocol works like Epidemic Routing [23]. They exchange summary vectors of their transport buffers. A summary vector identifies which messages a node contains on his buffer. This way, both nodes can find out which messages they do not share. Finally, they start sending messages.

The computation result is then used to decide if a message is stored in the transport buffer. When creating or receiving a message, our scheme uses a relevance function to decide if a message is stored in the transport buffer. Relevance function results are contained on the interval [0,1]. A low value indicates that a message has low relevance, and the probability of encounter with nodes interested in the message content is low. An high value represents a message with high relevance, and the probability of encounter with nodes interested in the message content is high.

A relevance function is a configurable parameter in easy-City. For example, a flooding-based protocol can be implemented by the following relevance function:

$$F_{rel}(m) = 1$$

ie, all messages are qualified with the same relevance.

A simple gossip-based [11] solution can be expressed by the following relevance function:

$$F_{rel}(m) = \begin{cases} 1 & : Random(0,1) > 1 - \beta \\ 0 & : Random(0,1) \leq 1 - \beta \end{cases}$$

where β is the probability of adding a message to the transport buffer.

Future-Location-Aware Relevance Function

It is possible to implement relevance functions that take into account information about node's mobility and interesting message destinations. An observation message has a pair of locations, or a segment, reporting travel or waiting times experienced during an user journey. Observation messages are particularly interesting at the source location (first location of the contained pair). We call the source location as message destination.

The following expression specifies a relevance function, which tries to take advantage of knowledge about future locations of the user:

$$F_{rel}(m) = \begin{cases} \frac{ttl_a(m) - (t(A,B) + t(B,d(m)))}{ttl_i(m)} \times \frac{MDEV - t(B,d(m))}{MDEV} \\ \text{if } ttl_a(m) \geq t(A,B) + t(B,d(m)) \& MDEV \geq t(B,d(m)) \\ 0, otherwise \end{cases}$$

where A is the node current location, B is the nearest location to the message destination (d(m)), from the node's future locations; MDEV is maximum tolerated travel time between B and d(m); $ttl_i(m)$ is the message's initial time-to-live; and $ttl_a(m)$ is the message current time-to-live. This relevance function takes into consideration the value of the message when it reaches the destination (first factor) and penalizes the deviation of the current node's path to the message destination (second factor).

3.3.2 Buffer Management

easy-City is meant to be executed on mobile devices (e.g, mobile phones) which, typically, have important physical memory restrictions. So, it is important to define appropriate buffer management policies. Transport buffers need to be maintained deleting old messages. The observation base and the transport buffer share the same data structure, so the detection and deletion of old messages removes the correspondent entries on the observation base.

It is important to specify what happens when a node runs out of buffer space. easy-City maintains transport buffers in the following way: when a transport buffer becomes full, the node drops the less relevant message. We adopt this mechanism because relevancy indicates the probability of timely delivering a message. This way, we try to minimize the impact of dropping a message to the system. If a node has several messages with the same relevancy, we employ a simpler approach, such as FIFO, Most-Forward of Less-TTL [4], as a tie-breaker.

3.3.3 Message Aggregation

Several nodes may observe and report the same event. So, these multiple reports originate similar messages. Similar messages contribute to a wasteful usage of network and node resources.

Hence, we developed a simple mechanism to detect and delete those similar messages. Two messages are similar if, and only if, they report the same locations pair and they have sufficiently close creation and experienced times.

easy-City message aggregation mechanism works as follows: i) When a node receives a message verifies if it contains a similar message on its buffer; ii) If the node has a similar message, it drops the received message; iii) Otherwise, it adds the message to its transport buffer.

3.3.4 Route Re-Computation

The basic solution, presented in Section 3.2, assumes that a node recomputes the best route option on each observation message reception. This strategy is potentially problematic on a realistic scenario because the searching algorithm is frequently executed even when no significant change has occurred. A more sensible approach consists on recomputing the best route option when a different location-time message (a location-time message that changes current node location) is received and the observation base has been changed.

4. EVALUATION

We now evaluate easy-City.

4.1 Methodology

In this paper, we simulated the execution of easy-City on the extended version of the ONE simulator [12]. We used HelsinkiMedium scenario provided with the simulator. We built a simulation scenario with 468 nodes running easy-City, consisting on 200 public transport users, 76 vehicles and 192 stations.

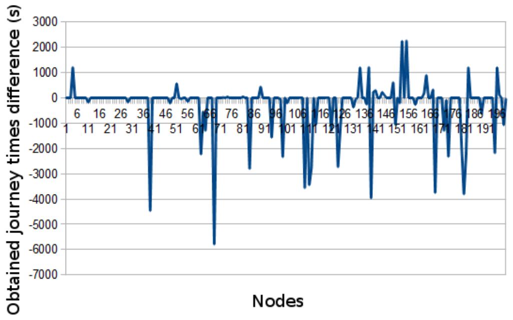
User nodes choose a random starting location and a random destination. They are organized in three groups. Group 1, with 50 users, starts moving on the starting instant of the simulation. Group 2, with 50 users, starts moving after 2200 seconds of simulation. Group 3, with 100 users, starts moving after 4200 seconds. Each node has a wireless interface for communication. In this simulation, we used a simple abstraction of Bluetooth Class 1 and 2 devices. We ran the simulation for approximately 4.5 hours on an area of approximately 8Km per 7Km. We used two operation modes of easy-City. Mode 1 consists on a single access to the map on the initial instant, when the users start the journey. This is the typical usage scenario of a journey planning application. Mode 2 consists on normal operation of easy-City, as previously explained. easy-City recomputes periodically the best route option, to keep the user informed of unpredictable events. easy-City is evaluated by comparing mode 1 and mode 2 operation in three distinct scenarios. The first scenario consists on an ideal scenario without traffic jams, accidents and service interruptions. The second scenario consists on a more realistic scenario with several traffic jams affecting services. We affect the fastest transport options to check the ability of user nodes to find better alternatives. The third scenario consists on a more problematic scenario where several services fail, due to service interruptions. In this scenario, we affect several services for a long period, to test the effectiveness of the infra-structure's warnings and the user nodes reaction. Our goal is to show that easy-City is particularly useful on a real-life scenario where unpredictable occurrences are common. On these evaluation scenarios, users always accept best route suggestions provided by easy-City.

4.2 Obtained Results

In the ideal scenario, we obtained identical results running both modes. As expected, static information provided by operators is similar or identical to the information captured by user nodes. So, the best route option obtained in the first access to the map is return in each map access performed by easy-City mode 2.

In the second scenario, easy-City mode 2 obtained better results than easy-City mode 1 in terms of travel times. easy-City mode 2 gives the users crucial information to avoid the negative impact of traffic jams on journeys, by suggesting alternative best route options. Figure 2 shows the obtained travel times difference comparing both modes. Negative values correspond to situations where nodes running mode 2 arrived earlier on their destination.

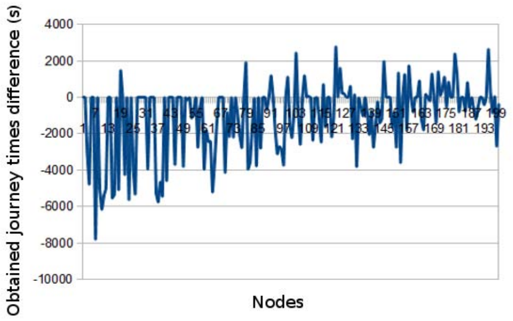
Figure 2: Difference between times obtained using mode 2 and times obtained using mode 1 on the second scenario



Analyzing the results we conclude that users running easy-City mode 2 arrived earlier on destination for 20% of journeys. In 70% of journeys, users observed similar travel times to mode 1 users. Finally, 10% of journeys, users running mode 1 arrived earlier.

In the third scenario, as we expected, easy-City obtained better results than easy-City mode 1. Users aided by easy-City mode 2 generally avoided the problematic services and finished their journeys earlier than users aided by easy-City mode 1. Figure 3 shows the obtained travel times difference comparing both modes.

Figure 3: Difference between times obtained using mode 2 and times obtained using mode 1 on the third scenario



In this scenario, users aided by easy-City mode 2 arrived earlier on destination 49% of journeys. 35% of journeys, they registered similar travel times to mode 1. Finally, 16% journeys, users running mode 1 arrived earlier.

5. CONCLUSIONS

In this paper we presented easy-City, a collaborative route search system for public transport users. easy-City establishes a delay tolerant network between mobile devices (e.g., mobile phones), carried by public transport users, and a simple infra-structure, installed on vehicles and on some stations, to disseminate information. We introduced easy-City's architecture and the evaluation of a developed implementation on a simulation environment. In the future, we will keep working on easy-City, with particular emphasis on exploring different usage scenarios to evaluate the system behaviour. Furthermore, we will proceed with the evaluation of the dissemination protocol described in this paper, comparing it with other popular solutions.

6. REFERENCES

[1] Shabbir Ahmed and Salil S. Kanhere. Hubcode: message forwarding using hub-based network coding in delay tolerant networks. In *Proceedings of the 12th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems*, MSWiM '09, pages 288–296, New York, NY, USA, 2009. ACM.

[2] Chicago Transit Authority. http://www.transitchicago.com/, 2010.

[3] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine. Maxprop: Routing for vehicle-based disruption-tolerant networks. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1 –11, april 2006.

[4] Hong-Tai Chou and David J. DeWitt. An evaluation of buffer management strategies for relational database systems. In *Proceedings of the 11th international conference on Very Large Data Bases - Volume 11*, VLDB '1985, pages 127–141. VLDB Endowment, 1985.

[5] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959. 10.1007/BF01386390.

[6] DTNRG. Delay tolerant networking research group. http://www.dtnrg.org/, 2011.

[7] Andreas Festag, Alban Hessler, Roberto Baldessari, Long Le, Wenhui Zhang, and Dirk Westhoff. Vehicle-to-vehicle and road-side sensor communication for enhanced road safety, 2008.

[8] Roy Friedman, Daniela Gavidia, Luis Rodrigues, Aline Carneiro Viana, and Spyros Voulgaris. Gossiping on manets: the beauty and the beast. *SIGOPS Oper. Syst. Rev.*, 41:67–74, October 2007.

[9] Wei Gao and Guohong Cao. User-centric data dissemination in disruption tolerant networks. In *INFOCOM, 2011 Proceedings IEEE*, pages 3119 –3127, april 2011.

[10] Google. Google transit. http://google.com/transit/, 2010.

[11] Zygmunt J. Haas, Joseph Y. Halpern, and Li Li. Gossip-based ad hoc routing. *IEEE/ACM Trans. Netw.*, 14:479–491, June 2006.

[12] Ari Keränen, Jörg Ott, and Teemu Kärkkäinen. The one simulator for dtn protocol evaluation. In *SIMUTools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, New York, NY, USA, 2009. ICST.

[13] A. Lindgren and K.S. Phanse. Evaluation of queueing policies and forwarding strategies for routing in intermittently connected networks. In *Communication System Software and Middleware, 2006. Comsware 2006. First International Conference on*, pages 1 –10, 2006.

[14] Anders Lindgren, Avri Doria, and Olov Schelén. Probabilistic routing in intermittently connected networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 7:19–20, July 2003.

[15] Bing Liu. Intelligent route finding: Combining knowledge, cases and an efficient search algorithm. In *In Proceedings of the 12th European Conference on Artificial Intelligence*, pages 380–384. Wiley and Sons, Ltd, 1996.

[16] Christian Lochert, Björn Scheuermann, Christian Wewetzer, Andreas Luebke, and Martin Mauve. Data aggregation and roadside unit placement for a vanet traffic information system. In *Proceedings of the fifth ACM international workshop on VehiculAr Inter-NETworking*, VANET '08, pages 58–65, New York, NY, USA, 2008. ACM.

[17] Transport For London. Journey planner. http://journeyplanner.tfl.gov.uk/.

[18] Transport For London. Tube. http://www.tfl.gov.uk/modalpages/2625.aspx.

[19] Ramesh Rajagopalan and Pramod K. Varshney. Data aggregation techniques in sensor networks: A survey. *Comm. Surveys & Tutorials, IEEE*, 8:48–63, 2006.

[20] RATP. Ratp site. http://www.ratp.fr/, 2010.

[21] Bluetooth SIG. The official bluetooth technology info site. http://www.bluetooth.com/, 2010.

[22] Thrasyvoulos Spyropoulos, Konstantinos Psounis, and Cauligi S. Raghavendra. Spray and wait: an efficient routing scheme for intermittently connected mobile networks. In *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, WDTN '05, pages 252–259, New York, NY, USA, 2005. ACM.

[23] Amin Vahdat and David Becker. Epidemic routing for partially-connected ad hoc networks. Technical report, 2000.

[24] Hui Ye, ZhiGang Chen, ZuoQun Xia, and Ming Zhao. A data dissemination policy by using human mobility patterns for delay-tolerant networks. In *Communications and Mobile Computing (CMC), 2010 International Conference on*, volume 1, pages 432 –436, 2010.

[25] Eiko Yoneki, Pan Hui, ShuYan Chan, and Jon Crowcroft. A socio-aware overlay for publish/subscribe communication in delay tolerant networks. In *Proceedings of the 10th ACM Symposium on Modeling, analysis, and simulation of wireless and mobile systems*, MSWiM '07, pages 225–234, New York, NY, USA, 2007. ACM.

[26] Ting Zhong, Bo Xu, and O. Wolfson. Disseminating real-time traffic information in vehicular ad-hoc networks. In *Intelligent Vehicles Symposium, 2008 IEEE*, pages 1056 –1061, june 2008.