CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2012; **00**:1–16
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cpe

A simulator for opportunistic networks

Cristian CHILIPIREA[1], Andreea-Cristian PETRE[1], Ciprian DOBRE[1*], Florin POP[1], George SUCIU[1]

[1]*University Politehnica of Bucharest, Splaiul Independentei 313, Bucharest, Romania*

SUMMARY

When mobile devices involved in a communication process are unable to establish a direct connection, or when communication should be offloaded to cope with large throughputs, mobile collaboration can be used to enable communication through opportunistic networks. These types of networks are formed when mobile devices communicate only using short-range transmission protocols, usually when users are close. Routes are built dynamically, since each mobile device is acting according to the store-carry-and-forward paradigm. Thus, contacts are seen as opportunities to move data towards the destination. In such networks the routing protocol is of vital importance and today we witness quite a number of routing algorithms that have been proposed to maximize the success rate of message delivery whilst minimizing the communication cost. Such protocols take advantage of the devices history of contacts, or information about users carrying the mobile devices, to make their forwarding decision. This paper extends our previous work with: First, we describe a new simplified, fast simulator, designed to minimize the work needed to conduct extensive tests for opportunistic routing algorithm on multiple traces; next we analyze extensively several of the most popular routing algorithms through extensive simulations conducted using our simulation platform. We highlight their pros and cons in different scenarios, considering different real-world mobility data traces, such as GPS traces. The raw GPS traces are converted to a format based on encounters between participating entities. Copyright © 2012 John Wiley & Sons, Ltd.

Received . . .

KEY WORDS: simulation; delay tolerant; opportunistic network; routing algorithm

1. INTRODUCTION

Mobile portable devices, such as smartphones, tablets, etc., or wearable devices, such as smart bracelets or smart watches are today more pervasive. The use of existing communication infrastructures is growing more and more as people can, and need, to connect to Internet now whenever and wherever. Our dependency on mobile technology has grown exceedingly. Still, there are several situations where we cannot extensively rely on traditional communication infrastructures, for instance antennas for broadband communications. These situations range from disasters scenarios, where existing communication gateways can be destroyed, to extreme locations (such as underground tunnels, remote places, where the costs of deploying communication infrastructures outweigh their benefits), and to situations involving crowds: mass events, such a parades, where a great number of users gathered in one place are trying to connect to a limited set of towers. Such situations increase the need to explore alternative means to manage the available wireless communication resources, to ensure the connection is always available when needed.

The emergence and wide-spread of new-generation mobile devices, together with the increased integration of wireless technologies such as Bluetooth and Wi-Fi, create, in fact, the premises for

*Correspondence to: University Politehnica of Bucharest, Spl. Independentei 313, Romania, E-mail: ciprian.dobre@cs.pub.ro new means of communication and interaction, challenge the traditional network architectures and are spawning an interest in alternative, ad-hoc networks such as *opportunistic mobile networks*.

An opportunistic mobile network (OMN) [33] is established in environments where human-carried mobile devices act as network nodes and are able to exchange data while in proximity. Whenever a destination is not directly accessible, a source would opportunistically forward data to (some of) its neighbors. The latter act as carriers and relay the data until the destination is reached or the messages expire.

To cope with unreliable connectivity and OMN partitioning, the natural approach is to extend the store-and-forward routing to store-carry-forward (SCF) routing [24]. In SCF routing, a next hop may not be immediately available for message forwarding. In this case, forwarding will be delayed until a suitable node is encountered. Thus, OMN nodes must be (i) capable of buffering data for a considerable duration, and (ii) selecting suitable carriers for a message, from the list of all sighted nodes. A bad forwarding decision may cause the packets to be delayed indefinitely [7].

In this context, given that mobile devices with increased capabilities (smartphones, tablets) and equipped with wireless communication capabilities continue to appear, would such devices be able to successfully carry messages destined for others, as envisioned by an opportunistic network model? As mentioned, OMNs are designed to support from catastrophic scenarios, where smartphones would take over the entire capacity of the damaged communication facilities in the disaster areas and beyond, all the way to completely distributed social networks, where mobile devices become caches of information in a publish/subscribe approach.

Experts predict that global mobile data traffic increased 26-fold between 2010 and 2015. At this growth a significant investment in the infrastructure will be needed. OMNs offer an alternative approach, because some of the wireless traffic can be offloaded directly into the mobile devices within the surrounding area. The current wireless communication infrastructures reached their threshold, as bottlenecks have already been observed (e.g., in 2009 a deployment of smartphones at a greater scale caused an overload in the AT&T infrastructure[†]). Will OMNs be of actual help in dispersing some of this overwhelming traffic directly to other smartphones? Are routing protocols for such challenged networks mature enough to be widely deployed?

In this paper we describe the CCPAC simulator, designed as a readily-available instrument for researchers and practitioners to evaluate their OMN approaches under a wide-range of conditions. It represents a standardization effort towards creating an instrument capable to assist developers in their evaluation, by minimizing the effort required to build and test a new routing algorithm. The paper extends our previous work [14] with: We make a critical analysis of the status regards the tools currently used for the evaluation of novel ad-hoc network paradigms. We compare our work with existing simulators, underlining key differences. We then analyze new case studies, comparing the most cited algorithms for routing in opportunistic networks over real-world traces collected in larger environments (city-wide scale evaluation). We detail our model and report on our findings on the use of the simulation tool with traces collected of vehicles in traffic. The raw GPS traces are converted to a format based on encounters between participating entities, suitable for the simulations presented in this paper. Finally, we extend the experimental result with new data sets and an updated discussion on the utility of our findings.

The rest of the paper is structured as follows. We first present work related to our own. Section 3 contains the OMN routing algorithms implemented in our simulator. Section 4 describes the simulator, followed in Section 5 by details on the mobility traces being used. Section 6 contains the analysis the obtained results. Finally, Section 10 concludes the paper.

2. RELATED WORK

There have been several attempts to build simulation platforms or frameworks in order to understand and extend network communication [34, 25, 26, 23, 21]. The framework described in [34] has OMNet++ [40] as a base, and provides simulated mobility patterns that can be used as input data

[†]http://www.nytimes.com/2009/09/03/technology

for various algorithms, which are the target of simulations. Our simulator is capable of using these mobility patterns as input traces, as we proved with the Random Waypoint trace we generated in a similar manner. OMNet++ [4], now at version 5, is probably one of the most popular network simulators. It provides full IDE support and an entire graphical interface in which simulations can be performed. Furthermore it can simulate the entire network stack and offers visualization on both communication and node movements using Open Street Map [6]. However because it has so many features and such a fine detail in simulation, it cannot scale to large number of devices communicating over large time periods of months or years. Our solution concentrates on only one thing, the routing process in opportunistic mobile networks. No other part of the network stack is simulated and no processing power is wasted on visualizations. Because of this we manage to parse and simulate routing algorithms on large traces in only a number of minutes. This permits us to evaluate multiple solutions and take into account multiple factors, such as energy efficiency, which we discussed in [11].

The Opportunistic Network Emulator, ONE Simulator [25, 26] accepts various movement models and event generators that can be used in a modular way as well as support for external DTN routing simulator. The visualization module is a powerful feature of this simulator, as it offers the possibility of visualizing result graphs and maps of the mobility patterns. In ONE the developer has to define different nodes for movement, persistency in data storage, energy consumption, etc. The ONE Simulator is no longer in development, with the last update made in 2013 [5]. Our simulator, CCPAC, offers the developer more flexibility, as he can define his own simulated models and entities.

This simulator permitted research in other directions. For instance [18] extended the ONE simulator in order to support self-organization methods in pervasive environments. This new simulator for instance directly supports GPS traces. This is also a need we felt for our solution and we implemented a GPS to encounters converter.

Some simulators, such as the ones used in [23] and [21] are developed for a specific set of algorithms and traces, thus they are not general. Furthermore we have not been able to find the software used for those works and development seems to have ended for those simulators.

Other simulators such as the ns-2 [27, 2] concentrate on specific parts of the network stack, even though they can be modified to work on others. For instance, ns-2, a simulator constructed for a DARPA project named Vint. The simulator comes paired with the nam [17] network visualizer, which can use the results of ns-2 to show how the packets move through the network.

More recently the ns-3 [20, 3] simulator is slowly trying to replace ns-2. The new simulator is written in pure C++ and outperforms ns-2 in things like memory management.

A better overview of network simulation tools is offered in [37]. In their article the authors compare 8 network simulators. However none of them is directly targeted at opportunistic networks. The results we show in this paper, for multiple algorithms, all have a low packet delivery rate. Even worse we have identified no scenario, and no algorithm that offers 100% delivery rate. This is mostly because there are nodes that never have contact with other nodes. In other words, regardless of time, there is no path in the encounters graph to get from any node to any other node.

Because of this low success rate in packet delivery there is no reason to simulate other parts of the network stack, such as TCP. TCP does support a large number of lost packets, but it does not support extremely high latencies or round trip times. In other words, TCP cannot sustain an opportunistic connection. We are not aware of any work in opportunistic networks that manage to successfully use TCP without major modification to the protocol.

Opportunistic networks can still be used in other scenarios such as message transmission (like e-mail) or data dissemination. Simulating only the routing decisions is enough to get a clear idea of the behavior of the protocols for this type of applications.

The need for a simplified simulator can be observed in works such as [9] where multiple opportunistic routing algorithms are compared using multiple simulators including specially constructed ones. The problem here is that because the algorithms are tested using different simulators which have different settings and possibly even different implementation errors, a direct comparison between the results of these different simulators is not exactly possible. When a simulator is designed for testing a specific algorithm it might show results that put that algorithm in the best light.

In order to offer an impartial solution, in this paper we test multiple algorithms on multiple traces. As to our knowledge, there is no work that compares these algorithms using the same simulator and the same methodology. Furthermore in this paper we do not include our solution for opportunistic routing in order to insist on the simulator and not to be biased towards one singular solution. Different approaches [19] do not use simulators of the routing algorithms. Instead they try to calculate the probability of packet exchanges taking place in different scenarios. This method does offer an idea of what to test the algorithms on, but does not offer the answer to what is the best solution.

3. OMN ROUTING ALGORITHMS

Various OMN routing algorithms exist, each of them created to address particular issues in opportunistic networks and with different data required for a proper functionality. In this section we described several of the most popular routing algorithms (the more cited ones, having made an impact in the related literature) for OMNs.

We mention here that some of the algorithms are purely academic. For example, Epidemic cannot be implemented in a real life scenario, as it requires unlimited storage capacity. Such algorithms are still very useful for comparisons; Epidemic together with Wait offer the lower and upper margins for our measurements.

In our simulator we implemented some of the most known OMN routing such as: Epidemic [39], The Wait routing algorithm (or Direct Delivery) [38], Multiple-Copy-multiple-hoP (or MCP) [22], dLife [29], Rank [22], Label [21], BUBBLE Rap A and B [22], PRoPHET Routing algorithm [28]. Al these algorithms are detailed at a greater length in our previous work [14]. Other algorithms such as PROPICMAN [31] and CiPRO [30] could not be implemented in our simulator, as they use context information that is not common to available mobility traces, such as the name and workplace of the device holder, or even hobbies.

4. THE CCPAC SIMULATOR

In our previous work we experimented with different OMN routing algorithms, in an attempt to include energy efficiency in the routing decision [11] or to use Gaussian processes to model encounters [12] between individuals that carry mobile devices. However, we have felt the need for standardized tools and platforms for testing our results. As such, we have built our OMN simulator called CCPAC, and continued to extend into a powerful evaluation instrument. The simulator allows us to concentrate on the routing algorithms and less on how to handle the packet transmission, reading the traces or how to make the simulation run faster.

The simulator, written in Java, can run all tests being proposed in [11, 12, 32]. It does not only offer a way to simulate OMNs (although this is now its main purpose) but as shown in [32], it can simulate any type of network and can even simulate scheduling of message transfers. And, unlike similar simulators (e.g., ONE, ns-3, etc.), CCPAC includes several optimizations that allow to experiment with large-scale scenarios, comprising potentially of thousands nodes.

To facilitate long term development, the simulator is built from a set of mod ules that can be easily interchanged (see Figure 1). These modules handle everything from message generation to time management. The most important module is probably the Node. A Node is the only part that needs to be changed to implement a new routing algorithm. It is fully configurable by the user so that new features can be added as needed. In [11] we configured the Node for a number of known routing algorithms to add support for Energy Awareness; the devices would have a limited battery resource.

Currently the simulator supports only ideal transmission medium (all packets are sent successful and there are no delays). We leave packet loss for future work.

The CCPAC simulator is composed of several modules:

- ***Connection*** handles the data of a connection (two nodes, start and end time of the encounter)
- ***Input*** reads the trace inputs and parses them; the result should always be a NodeGraph. Input can be used to generate a trace. The simulator is built to function with mobility traces that offers a list of two or more device detections with timestamps.
- ***NodeGraph*** stores the data required to run the simulation and it handles the message transfers.
- ***Message*** Represents a message and its payload. A message can be split into Packets based on the size of the message. In the experiments presented here the message size is 1, but larger sizes can be used if partial data transfers are in the scope of the analysis.
- ***MessageGenerator*** Generates all the messages in the network, it is called at every simulation step and it decides if more messages are to be be generated or not. For the results presented here we used a generator which splits the simulation time in 20 steps and generates 2 messages per node at each step with a randomly selected destination.
- ***Node*** represents a single device in the network. A Node needs to have *initialize()*, *addPacket()* and *removePacket()* as well as *prepareSendPackets()* implemented. The last function should handle the transfer of Packets to its current neighbors and call the NodeGraph for the transfer.
- ***Packet*** represents a part of a Message.
- ***PrintStatistics*** outputs any required statistics, either from the Node or from the Messages.
- ***Scheduler*** Splits a Message into Packets and makes them available to the Nodes at the time they are allowed to be transferred. It was used in [32] to schedule Virtual Machine Transfers between physical machines inside hybrid clouds.
- ***SimulationTime*** handles the time in the simulation. Two types of time passage were implemented by us: *real time* (each second is processed) and *event based time* (events are processed). The event based simulation is more efficient as inactivity periods are skipped.

The above modules were designed to offer a simple, configurable simulation environment. A standard simulation follows the flow shown in Figure 2. Here we can see that most of the work is executed by the Node Graph (NG) and each Node executes its own code. At the end of an execution step the simulation time is recalculated. The simulation continues until either the simulation time ends or there are no more events to be simulated.

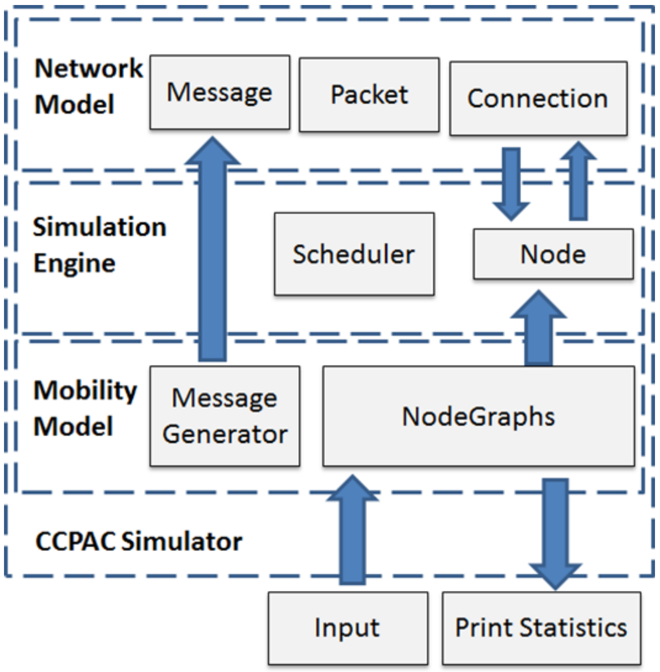

Figure 1. Architecture of the simulator.

There are a number of improvements designed to make the simulator run more efficiently. For example, instead of simulating every second or every tick, if we want to have a different resolution, from the start to the end of the trace the simulator jumps to events if no message transfer was done in the last tick. Events represent any change to the network topology. The NodeGraph manages efficiently the lists of all the neighbors of all the nodes that are currently connected.

Another improvement was in the way of handling packets, instead of building a different object for every transfer only the reference to the object changes.

The process of programming a new routing algorithm is detailed in [14].

5. TRACES USED IN OUR EXPERIMENTS

For our analysis of OMN algorithms five different publically available, real-world mobility data traces were used. We also executed the algorithms on a synthetic trace. The experimental period ranges from a few days (as in case of Infocom05) to almost a year (in case of Reality). Traces include both internal and external devices; the internal ones have connections more often and are carried by individuals that are part of the experiment. For the real-life traces only the internal devices were used in order to limit external interference. The following traces were used:

- The Reality [16] trace consists of contacts logged from 100 smart phones preinstalled with software that considered contacts over Bluetooth at an interval of 5 minutes. Out of the total of 100, 75 users are either students or faculty staff in the MIT Media Laboratory, while the remaining 25 are students to an adjacent faculty. The study contains data spread over a period of 9 months.
- The Infocom05 [36] trace includes Bluetooth sightings by groups of users carrying small devices (iMotes) for three days during the Conference IEEE Infocom in Grand Hyatt Miami.

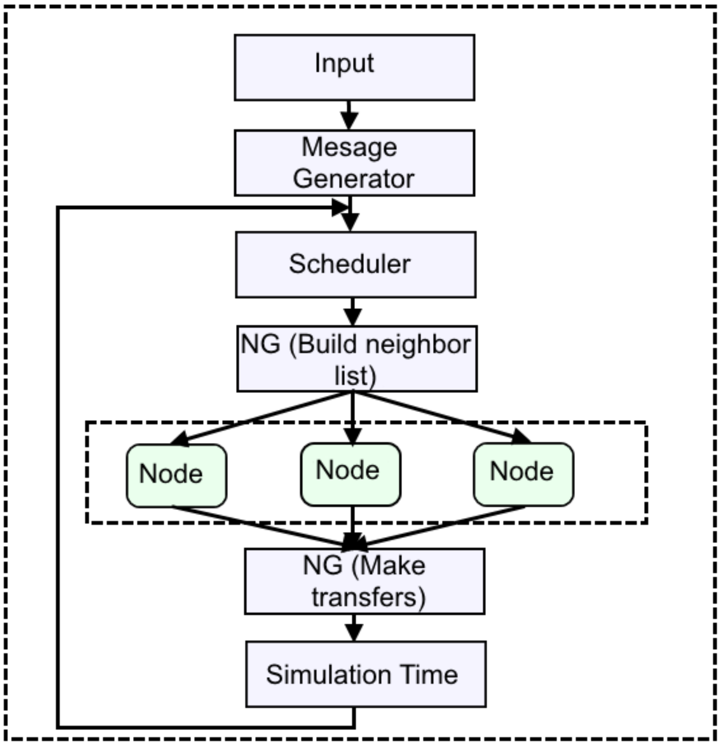

Figure 2. Simulator workflow.

Experimental data sets	Duration (days)	Number of Internal Experimental Devices	Number of Internal Contacts
Reality	246	97	113,367
Infocom05	3	98	22,459
Random	30	30	2,871

Table I. Characteristics of the Experimental Data Sets.

More specifically, the device users were students attending the student workshop. 41 devices recorded Bluetooth encounters starting from March 7th, 2005 and until March 10th, 2005.

- Random waypoint mobility trace is similar to the one used in [34]. It is a simulated trace with a uniform distribution. The encounters were recorded every 60 seconds and the trace lasts 30 days. There are 30 nodes that are considered. In the case of Random the number of nodes or days available is easily configurable, but we run all the tests with the same generated trace.

Table I summarizes the main features of the traces we present in this paper. In addition to these traces the Simulation Platform also offers support for Infocom06, UPB2011 and UPB2012. Adding new traces to the platform is simple because of the integrated parser provided as well [11, 12, 15].

6. SIMULATION RESULTS

This section presents the results obtained using the described simulator, on the following traces: Reality, Infocom05 and Random waypoint. We believe that this set of traces cover most of the possible scenarios.

As it can be observed in Figures 3 to 8, we used two metrics that are most relevant for analyzing an OMN routing algorithm: the Delivery Ratio and the Total Cost. We did not add any routing table information because most of the presented algorithms do not use it at all and for lack of space.

Delivery Ratio represents the percentage of messages that reach their destination and its metric is defined for a specific trace over a time period:

$$Delivery\ Ratio(trace, \Delta t) = \frac{No.\ of\ Received\ Messages(trace, \Delta t)}{Total\ No.\ of\ Sent\ Messages(trace, \Delta t)} \tag{1}$$

Total Cost represents the total number of transfers divided by the number of messages that were to be transmitted metric is defined for a specific trace over a time period, for all packets exchanged:

$$Total\ Cost(trace, \Delta t) = \frac{Total\ No.\ of\ Transfers(trace, \Delta t)}{Total\ No.\ of\ Sent\ Messages(trace, \Delta t)} \tag{2}$$

In case of Reality (Figures 3 and 4), we observe a high delivery ratio for most algorithms. Even the Wait algorithm has a delivery ratio of 0.4. This is caused by the considerable length of time, almost one year, that characterizes the Reality trace and, as such most nodes will eventually encounter most of all other nodes.

Unlike the Reality trace, the Infocom05 trace (Figures 5 and 6), spans over a period of 3 days (the duration of the Infocom 2005 conference). Because of this, no increase of the TTL passed the 3 day margin affects the results in any way. In this case we also noticed that even though the delivery ratio is almost the same for all of the included algorithms, with the exception of Wait, costs present a clear distinction. This phenomenon is caused by the large number of connections in the Infocom trace: over the span of the 3 days, the nodes are in contact with each other at a very high rate. This trace manages to clearly differentiate the algorithms, as the transmission rate in a crowded environment varies greatly from one to another. We believe this trace is a clear indicator why BUBLE Rap is such a popular algorithm.

Random waypoint (Figures 7 and 8), is a synthetic trace we constructed to be used as a baseline in our comparisons. The results are very similar with the ones obtained from Infocom 05, and this

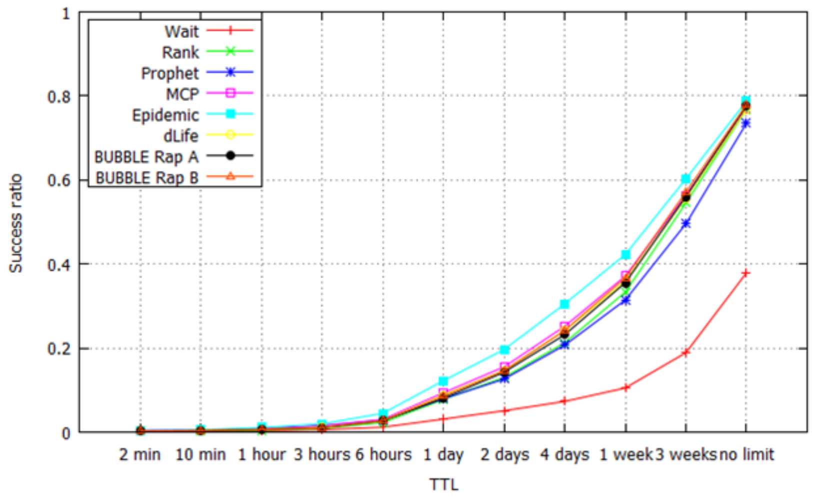

Figure 3. Reality Delivery Ratio.

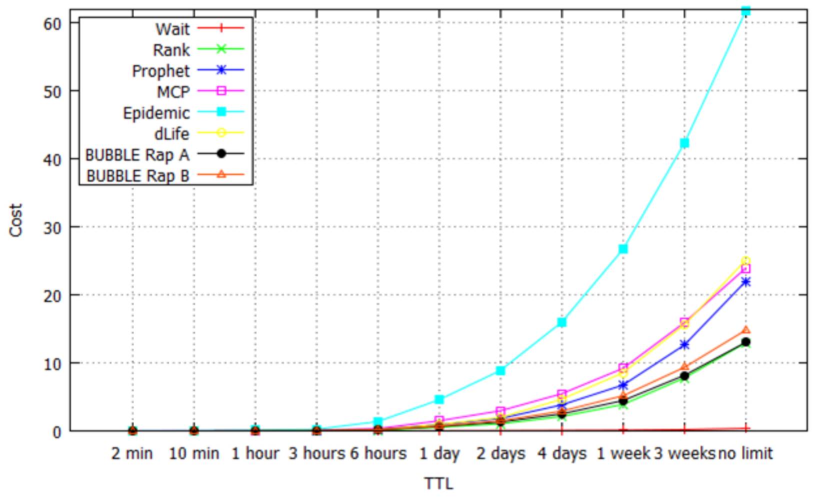

Figure 4. Reality Total Cost.

confirms the correctness of both the implementation of the algorithms as well as the fact that the Random Waypoint approach offers a simulated environment close to the one we find in real life.

Figure 9 represents the latency as mean number of hops for all delivered messages for all the algorithms on all the traces. It shows the expected behavior of each algorithm for successful delivery of messages. For instance we can see how Wait, as it is designed only has 1 hop for all successfully delivered messages.

In comparison to Figure 9, in Figure 10 we show latency in time as a mean over all delivered messages. Because the time intervals of the traces differ so much we had to scale down the Reality trace by dividing all values by 10, and scale up the Infocom 05 trace by dividing all values by 10. Comparing the 2 figures we can better understand the algorithms behaviors. For instance looking at the second picture we can see that Wait has an extremely high latency even if the number of hops is always 1.

A correct analysis of these algorithms takes a lot of factors into account. We haven't shown here information such as battery status, memory status or routing table. Our Simulator supports multiple Print Statistics modules and configurable Nodes that could show all kinds of information about devices or transmitted packets.

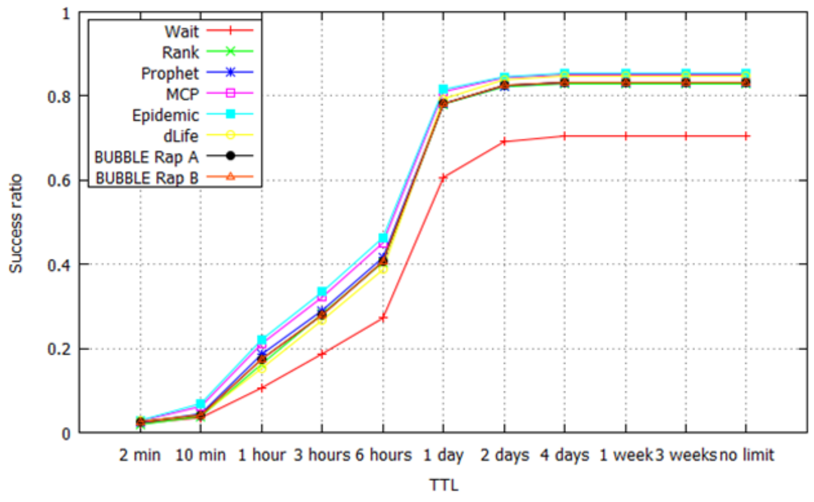

Figure 5. Infocom05 Delivery Ratio.

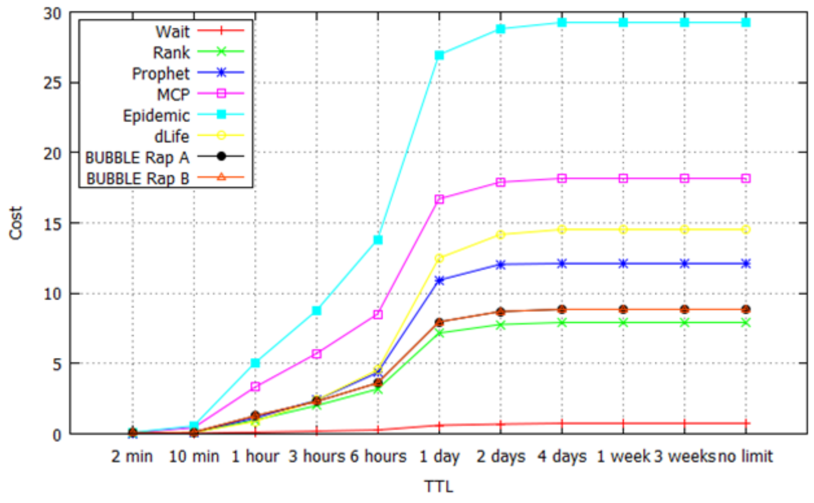

Figure 6. Infocom 05 Total Cost.

We have noticed with our traces that three algorithms manage to outperform the rest in most scenarios. These algorithms, Rank and BUBBLE Rap A/B are also the most popular ones. We add that even though BUBBLE Rap outperforms Rank in most cases, Rank proves to be a very powerful algorithm. Rank is one of the simplest algorithms to implement and it bases its forwarding decision only on a single global statistics, the popularity of the node. Because of the difficulties in obtaining community data we believe that Rank might prove to be the best choice in the decision regarding 'What delay tolerant routing algorithm is worth implementing?'.

7. MODELING THE TRACES

An encounter trace represents a set of graphs where every vertex is a device and an edge is an encounter between two different devices. Because the encounters change in time, the graphs all have the same set of vertices, but the edges permanently change. If all devices were in range of each other at all times then the graphs would all be full meshes.

We define the graphs as:

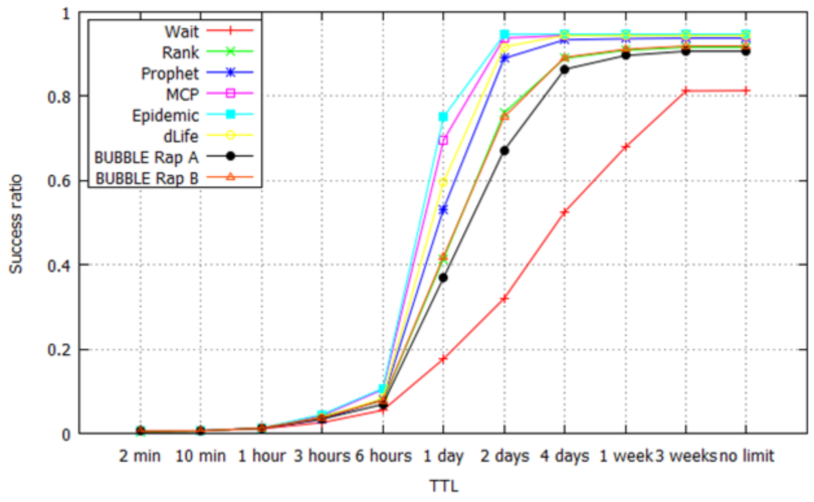

Figure 7. Random Waypoint Delivery Ratio.

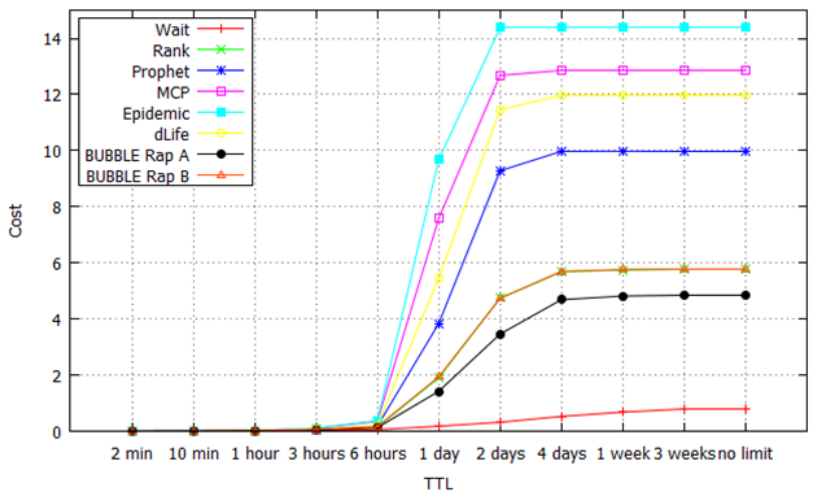

Figure 8. Random Waypoint Total Cost.

$$G(t) = (V, E(t))$$

Here:

$$V = \{v_i | v_i \textit{ is a device carried by an individual}\},$$

$$E(t) = \{e_{ij} = (v_i, v_j) | v_i, v_j \in V \textit{ and } v_i \textit{ is in detection range to } v_j \textit{ at time } \mathrm{t}\}$$

An encounter trace includes all edges followed by a time stamp, indicating the different graphs.

The GPS data sets include the ID of the device, equivalent to the vertices v_i in the previous model, followed by the location, in latitude and longitude, as well as a time stamp. Unlike the encounters case, a missing entry at a certain time stamp does not mean the device no longer exists. Usually entries are entered every few seconds, when the device updates its location using the GPS sensor data.

GPS data entries take the following form:

$$< ID; latitude; longitude; time >$$

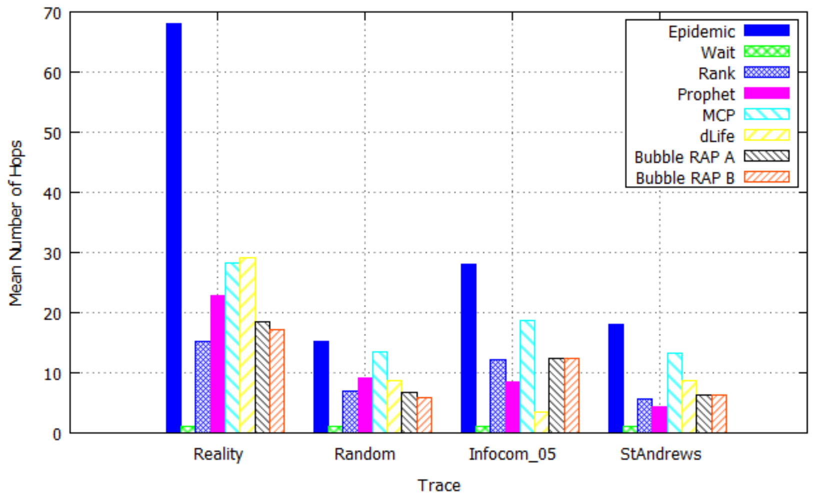

Figure 9. Mean Number of hops for delivered messages.

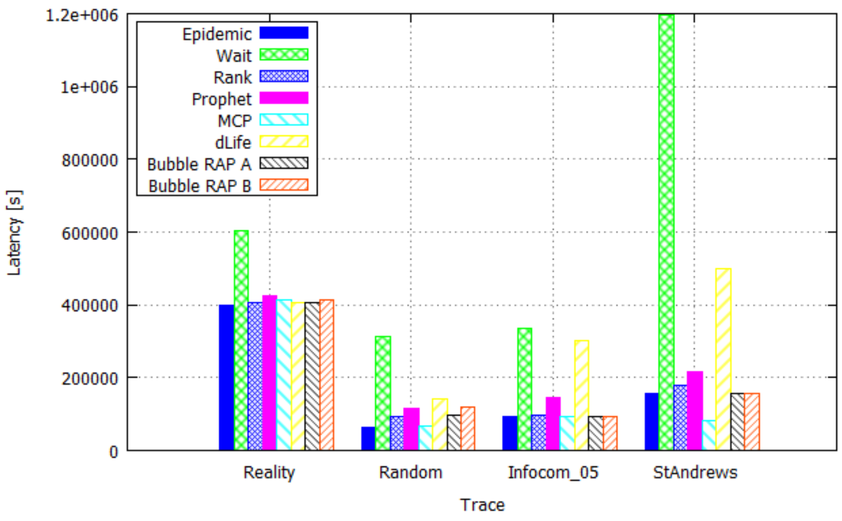

Figure 10. Latency of delivered messages for all algorithms.

Using latitude and longitude and a theoretical model of the detection range we can use this type of data to determine when encounters between two different devices take place.

In order to compensate the missing entries between time t and $t+1$ we approximate the time at 1 minute intervals. Because detections usually take place every few seconds, a one minute approximation should be enough to insure that no recorded detections are missed.

In the following chapter we explain how we convert GPS traces to encounter traces based on the models described above.

8. GPS DATA SETS

People behave differently depending of their daily activities and this leads to different behaviors of the crowds they are part of. For instance during week days people have regular hours in which they go and return from work. When a special event, such as a concert is organized in town, these activities and the patterns they produce generally change. A lot of people participate in the event

	Roma	San Francisco
# of vehicles	316	536
# of entries	21.196.198	10.262.204
# of encounters	864.566	7.900.352
Year	2014	2008

Table II. Vehicular data sets comparison.

rather than following their normal routine, while others are forced to go around the event area because of road blocks.

In order to be able to understand different opportunistic solutions and to see the behavior of the network, it is esential to test these solutions on large number of data sets that express different behaviors of crowds. The variance in crowd behavior may influence the results. For instance, the storage time should differ from a high individual density scenario to a low density one. Furthermore, being able to reliably test opportunistic communication opens the way to other applications such as mobile clouds [13], which are highly dependent on communication latencies.

Currently the literature does not offer many options with regard to different data sets that show encounters between individuals carrying WiFi enabled mobile devices. A database containing these type of traces and similar ones is provided by crowdad [1]. With the limited number of potential real life traces that represent encounters, there is a need to create more traces.

Creating a trace is difficult, it requires specialized devices, different scenarios and a large number of volunteers willing to carry devices that record detections. However we have identified that there are also available traces where car [8, 35] and pedestrian movements are tracked. These traces are made using GPS enabled devices that are carried by individuals. There are not many such traces publically available, but by converting this traces in the type we need and using them in conjunction with the ones made specifically for our purpose gives us more options and scenarios with which to test our solutions.

In order to convert GPS traces to encounters between individuals we need to use a model of the communication range at which we expect the devices to be able to interact. Different technologies, like radio, WiFi or Bluetooth have different ranges at which, in ideal scenarios, communication is possible. Because we use GPS data that only offers latitude and longitude, we model the encounter range as a disc around the device. We chose to use a range for the model that matches the middle range solution, WiFi. This means we consider an encounter any moment when two devices are at less than 100m from each other.

The data sets for GPS recording usually have recordings for every few seconds. Devices are not synced, this means encounters could be missed if we take all positions of devices in the same second and they are recorded in slightly different ones. In order to obtain correct encounters we split the time in 60 seconds intervals and compare all positions in this interval. We determined empirically that 60 seconds is an optimal solution. It corresponds with the 5-7 second difference between devices and it is small enough to assume the devices did not change their position by a large margin.

Before we made the conversion we filtered duplicates out of the data sets and all data items with GPS positions outside the grid of their respective cities.

We apply our solution for two distinct data sets. These data sets represent traces taken from taxi cars that carried around devices that tracked their movements using GPS. The traces were generated using taxis from Roma [8] and San Francisco [35]. Both traces expanded over a period of a month a contained data from hundreds of taxis. Table II shows a direct comparison between the two data sets. We can see that even if Roma has the largest original data set, after we convert it to encounters, it is almost ten times smaller than the San Francisco one. This is because there are more taxis in San Francisco being tracked and these taxis are very likely to return and meet at a taxi cab garage in the middle of the city.

The data sets are years apart and they show the behavior of individuals in the city. A more in depth comparison between the two data sets can be observed in [10].

9. VEHICULAR DATA SET EXPERIMENTAL RESULTS

To further explore the capabilities of the simulator we extended it with the ability to convert GPS traces to encounter traces, using the methodology we described in the previous chapter. Except of the module that converts the data sets we have made no other modification to the simulator itself or the algorithms that we considered.

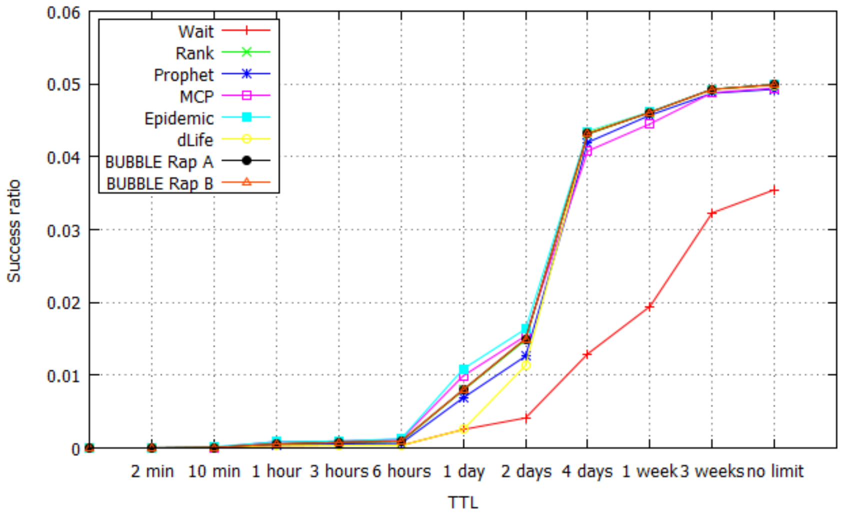

Figure 11. Roma delivery ratio

We compare the results for the vehicular data sets in the same manner we previously compared the results for the reality, random, StAndrews and Infocom trace. In Figure 11 one can observe the delivery ratio for the Roma data set. First of all, the maximally obtained delivery ratio is far smaller than the delivery ratio in the other data sets. Furthermore the difference between the different algorithms is not as clear as it previously was.

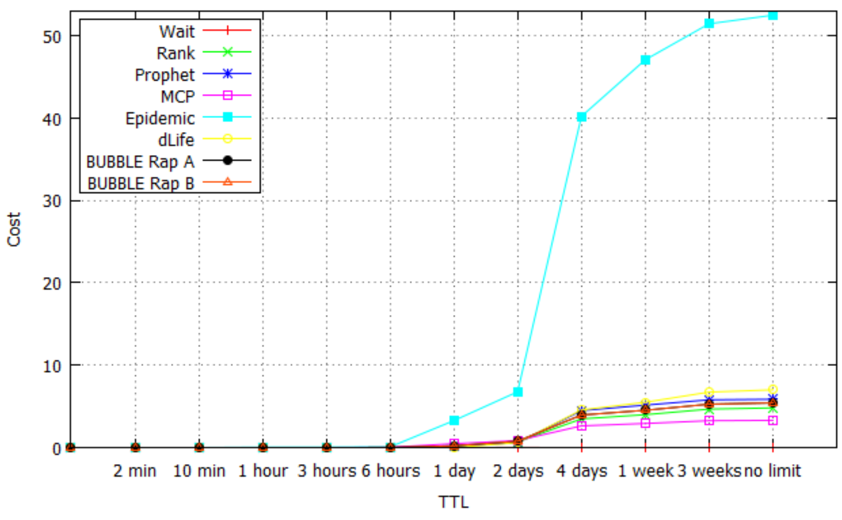

Figure 12. Roma Total Cost

When we look at total cost for Roma, Figure 12, Epidemic clearly uses far more packet transmissions then all others. This is curios considering the delivery ratio is still very similar with the other algorithms. Furthermore, unlike the previous data sets Bubble RAP solutions no longer stand out, they actually have results that are worse than others.

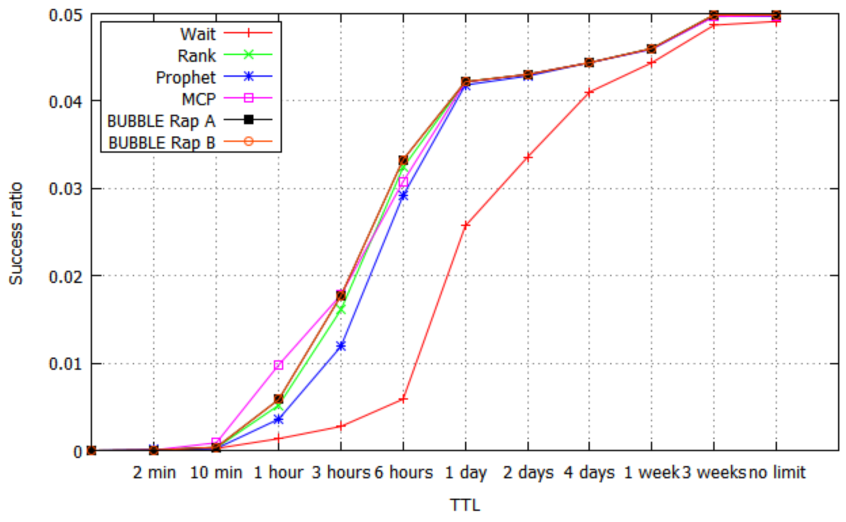

Figure 13. San Francisco delivery ratio

Looking at the results for San Francisco they are very similar with the ones for Roma, for instance Figure 13, showing the percentage of packets that were successfully delivered to their destination has extremely low results. Under all circumstances the delivery ratio is under 5% and the algorithms, given enough time, are not far from the Wait algorithm, the lower boundary.

When we look at total cost for the San Francisco trace, Figure 14, we can clearly observe a difference between the numbers of packet exchanges the algorithms use. To our surprise Bubble RAP had the highest total cost even though it did not have a higher delivery ratio compared to the other solutions.

We did not include the results for dLife and Epidemic because the number of packet exchanges for delivered packet, the total cost, is far greater than that of the other algorithms. dLife for instance goes beyond 300 packets.

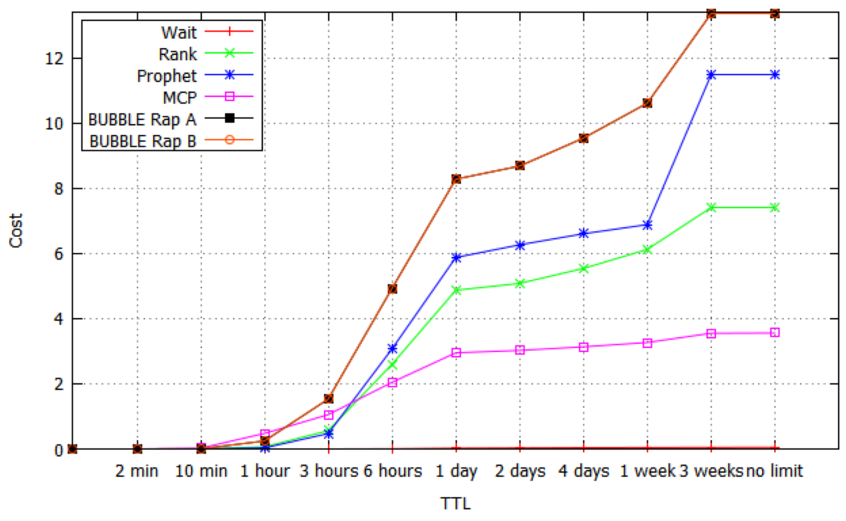

Figure 14. San Francisco Total Cost

With these vehicular traces we show how the algorithms behave completely different than expected when the number of devices is low and the area in which they move is the size of a large city without devices regularly meeting in one central location like is the case for reality or StAndrews.

Furthermore with these results we show that depending on the number of devices in the network and the diversity of encounters between them there might be a need to change the routing algorithm being used. A possible solution could even be to use the existing infrastructure such as 3G when we detect the diversity in encounter density in the network is really low

Comparing these results with the previous one we would like to clarify that the number of encounters, the density itself is not such an issue as the diversity of this encounters is. For instance if two devices continuously meet during a day (like is the case for two people working together) they do not contribute very much to the overall network if they dont produce encounters with other devices. The difficulty here being that diversity of encounters at network scale is more difficult to detect than the density of encounters.

10. CONCLUSIONS

In this paper we described a new Opportunistic Network simulator. Its purpose is to minimize the effort required to implement new algorithms to a minimum, and allows the developers to concentrate solely on the algorithm instead of a simulator or a platform. The simulator itself is available on-line, on github, and free to use.

With the results we obtained, we demonstrated the functionality and correctness of the simulator.

Still there are a number of improvements that are needed. First, we will add more traces to the ones already included; this will offer a more detailed view of the intricacies of the algorithms that are tested. Next we need to add other state of the art algorithms as they are created, this will offer a simpler way to compare new developments with other high performers.

ACKNOWLEDGEMENT

The research in this paper is supported by national project MobiWay - Mobility beyond Individualism (PN-II-PT-PCCA-2013-4-0321), and by national project DataWay - Real-time Data Processing Platform for Smart Cities: Making sense of Big Data (PN-II-RU-TE-2014-4-2731). The authors would like to thank the reviewers for their time and expertise, constructive comments and valuable insight.

References

1. Crawdad - `http://crawdad.org` (accessed: 2015-11-20).
2. The network simulator - ns-2 - `http://www.isi.edu/nsnam/ns/` (accessed: 2015-11-20).
3. ns-3 - `https://www.nsnam.org` (accessed: 2015-11-20).
4. Omnet++ - discrete event simulator - `https://omnetpp.org/` (accessed: 2015-11-20).
5. The one - `https://www.netlab.tkk.fi/tutkimus/dtn/theone/` (accessed: 2015-11-20).
6. Openstreetmap - `https://www.openstreetmap.org/` (accessed: 2015-11-20).
7. Fredrik Bjurefors, Per Gunningberg, Christian Rohner, and Sam Tavakoli. Congestion avoidance in a data-centric opportunistic network. In *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*, pages 32–37. ACM, 2011.
8. Lorenzo Bracciale, Marco Bonola, Pierpaolo Loreti, Giuseppe Bianchi, Raul Amici, and Antonello Rabuffi. CRAWDAD dataset roma/taxi (v. 2014-07-17). Downloaded from http://crawdad.org/roma/taxi/20140717, July 2014.
9. Eyuphan Bulut. *Opportunistic routing algorithms in delay tolerant networks*. PhD thesis, Rensselaer Polytechnic Institute, 2011.
10. C. Dobre F. Pop F. Xhafa C. Chilipirea, A-C. Petre. Enabling vehicular data with distributed machine learning. In *accepted for publication in TCCI journal*, 2015.
11. Cristian Chilipirea, A Petre, and Ciprian Dobre. Energy-aware social-based routing in opportunistic networks. In *Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on*, pages 791–796. IEEE, 2013.
12. Cristian Chilipirea, Andreea-Cristina Petre, and Ciprian Dobre. Predicting encounters in opportunistic networks using gaussian process. In *Control Systems and Computer Science (CSCS), 2013 19th International Conference on*, pages 99–105. IEEE, 2013.

13. Cristian Chilipirea, Andreea-Cristina Petre, Ciprian Dobre, and Florin Pop. Enabling mobile cloud wide spread through an evolutionary market-based approach. 2015.
14. Cristian Chilipirea, Andreea-Cristina Petre, Ciprian Dobre, Florin Pop, and George Suciu. A simulator for analysis of opportunistic routing algorithms. In *Parallel and Distributed Computing (ISPDC), 2015 14th International Symposium on*, pages 27–36. IEEE, 2015.
15. Radu Ioan Ciobanu, Ciprian Dobre, and Valentin Cristea. Social aspects to support opportunistic networks in an academic environment. In *Ad-hoc, Mobile, and Wireless Networks*, pages 69–82. Springer, 2012.
16. Nathan Eagle and Alex Pentland. Reality mining: sensing complex social systems. *Personal and ubiquitous computing*, 10(4):255–268, 2006.
17. Deborah Estrin, Mark Handley, Mark H, John Heidemann, Steven Mccanne, Ya Xu, and Haobo Yu. Network visualization with the vint network animator nam. Technical report, 1999.
18. J.L. Fernandez-Marquez, F.L. De Angelis, G. Di Marzo Serugendo, G. Stevenson, and G. Castelli. The one-sapere simulator: A prototyping tool for engineering self-organisation in pervasive environments. In *Self-Adaptive and Self-Organizing Systems (SASO), 2014 IEEE Eighth International Conference on*, pages 201–202, Sept 2014.
19. J. Gebert and R. Fuchs. Probabilities for opportunistic networking in different scenarios. In *Future Network Mobile Summit (FutureNetw), 2012*, pages 1–8, July 2012.
20. Thomas R Henderson, Mathieu Lacage, George F Riley, C Dowell, and JB Kopena. Network simulations with the ns-3 simulator. *SIGCOMM demonstration*, 14, 2008.
21. Pan Hui and Jon Crowcroft. How small labels create big improvements. In *Pervasive Computing and Communications Workshops, 2007. PerCom Workshops' 07. Fifth Annual IEEE International Conference on*, pages 65–70. IEEE, 2007.
22. Pan Hui, Jon Crowcroft, and Eiko Yoneki. Bubble rap: Social-based forwarding in delay-tolerant networks. *Mobile Computing, IEEE Transactions on*, 10(11):1576–1589, 2011.
23. Arshad Islam and Marcel Waldvogel. Reality-check for dtn routing algorithms. In *Distributed Computing Systems Workshops, 2008. ICDCS'08. 28th International Conference on*, pages 204–209. IEEE, 2008.
24. Evan PC Jones and Paul AS Ward. Routing strategies for delay-tolerant networks. *Submitted to ACM Computer Communication Review (CCR)*, 2006.
25. Ari Keränen and Jörg Ott. Increasing reality for dtn protocol simulations. *Helsinki University of Technology, Tech. Rep*, 2007.
26. Ari Keränen, Jörg Ott, and Teemu Kärkkäinen. The one simulator for dtn protocol evaluation. In *Proceedings of the 2nd international conference on simulation tools and techniques*, page 55. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009.
27. Sandeep Bajaj Lee, Eep Bajaj, Lee Breslau, Deborah Estrin, Kevin Fall, Sally Floyd, Padma Haldar, Mark H, Ahmed Helmy, John Heidemann, Polly Huang, Satish Kumar, Steven Mccanne, and Haobo Yu. Improving simulation for network research. Technical report, University of Southern California, 1999.
28. Anders Lindgren, Avri Doria, and Olov Schelén. Probabilistic routing in intermittently connected networks. *ACM SIGMOBILE mobile computing and communications review*, 7(3):19–20, 2003.
29. Waldir Moreira, Paulo Mendes, and Susana Sargento. Opportunistic routing based on daily routines. In *World of wireless, mobile and multimedia networks (WoWMoM), 2012 IEEE international symposium on a*, pages 1–6. IEEE, 2012.
30. Hoang Anh Nguyen and Silvia Giordano. Context information prediction for social-based routing in opportunistic networks. *Ad Hoc Networks*, 10(8):1557–1569, 2012.
31. Hoang Anh Nguyen, Silvia Giordano, and Alessandro Puiatti. Probabilistic routing protocol for intermittently connected mobile ad hoc network (propicman). In *World of Wireless, Mobile and Multimedia Networks, 2007. WoWMoM 2007. IEEE International Symposium on a*, pages 1–6. IEEE, 2007.
32. Mihaela-Catalina Nita, Cristian Chilipirea, Ciprian Dobre, and Florin Pop. A sla-based method for big-data transfers with multi-criteria optimization constraints for iaas. In *Roedunet International Conference (RoEduNet), 2013 11th*, pages 1–6. IEEE, 2013.
33. Luciana Pelusi, Andrea Passarella, and Marco Conti. Opportunistic networking: data forwarding in disconnected mobile ad hoc networks. *Communications Magazine, IEEE*, 44(11):134–141, 2006.
34. Agoston Petz, Justin Enderle, and Christine Julien. A framework for evaluating dtn mobility models. In *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, page 94. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009.
35. Michal Piorkowski, Natasa Sarafijanovic-Djukic, and Matthias Grossglauser. CRAWDAD dataset epfl/mobility (v. 2009-02-24). Downloaded from http://crawdad.org/epfl/mobility/20090224, February 2009.
36. James Scott, Richard Gass, Jon Crowcroft, Pan Hui, Christophe Diot, and Augustin Chaintreau. Crawdad trace cambridge/haggle/imote/infocom (v. 2006-01-31). *Download from http://crawdad. cs. dartmouth. edu/cambridge/haggle/imote/infocom*, 2006.
37. S Siraj, A Gupta, and R Badgujar. Network simulation tools survey. *International Journal of Advanced Research in Computer and Communication Engineering*, 1(4):199–206, 2012.
38. Thrasyvoulos Spyropoulos, Konstantinos Psounis, and Cauligi S Raghavendra. Single-copy routing in intermittently connected mobile networks. In *Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004. 2004 First Annual IEEE Communications Society Conference on*, pages 235–244. IEEE, 2004.
39. Amin Vahdat, David Becker, et al. Epidemic routing for partially connected ad hoc networks. Technical report, Technical Report CS-200006, Duke University, 2000.
40. András Varga and Rudolf Hornig. An overview of the omnet++ simulation environment. In *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, page 60. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.