A Simulator for Analysis of Opportunistic Routing Algorithms

Cristian Chilipirea, Andreea-Cristina Petre, Ciprian Dobre, Florin Pop, George Suciu
University POLITEHNICA of Bucharest
Bucharest, Romania
E-mails: cristian.chilipirea@cs.pub.ro, andreea.petre@cti.pub.ro, ciprian.dobre@cs.pub.ro, florin.pop@cs.pub.ro, george@beia.ro

Abstract

When mobile devices are unable to establish direct communication, or when communication should be offloaded to cope with large throughputs, mobile collaboration can be used to facilitate 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, can help applications exchange data. 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. Our contribution in this paper is two-fold: First, we present a new simplified, fast simulator, designed to minimize the work needed to conduct extensive tests for opportunistic routing algorithm on multiple traces; next we present an extensive analysis of 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.

Keywords: simulation, delay tolerant networking.

1. Introduction

Mobile portable (smartphones, tablets, etc.) or wearable devices (smart bracelets, watches, etc.) are today more pervasive. We are witnesses to a tremendous increase in the use of existing communication infrastructures, 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 (i.e., antennas for broadband communications), ranging from disasters scenarios (catastrophic events can lead to existing communication gateways being destroyed), to extreme locations (i.e., underground tunnels being explored, in remote places such as jungles, deserts, where the costs of deploying communication infrastructures outweigh their benefits), or situations involving crowds (in mass events, such a parades, a great number of users concentrating in one place is 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 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) [23] 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 intermittent 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 [11].

Routing algorithms for OMNs are either mobility-aware or social-aware. The former (which includes protocols such as PRoPHET [14]) takes routing decisions based on the number and duration of node encounters; the latter (which includes BUBBLE Rap [9]) relies on the knowledge that members (or nodes) of an OMN are people carrying mobile devices. Social-aware OMNs involve routing decisions based on how people are organized into communities, according to places of living and work, common interests, leisure activities, etc.

In this context, given that more powerful mobile devices (smartphones, tablets) 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 an overload in the AT&T infrastructure due to wider deployment of smartphones was reported[1]). 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 fact, even though today we are witnessing an abundance of OMNs routing algorithms for SCF being proposed, each one seems to support a particular mobility scenario: in various papers, authors validate their algorithms either for conference environments or city scenarios, or are maximizing one particular communication feature (i.e., such as delivery rate). We are witnessing a lack of a standardization methodology for comparing these routing algorithms available in the literature. This aspect is further complicated by the fact that different approaches rely on special requirements (i.e., there are algorithms that are based on the use of social data, somehow being already available from Facebook; others, like Ciproc and Propicman, require the use of detailed information about individuals that are part of the network, disregarding the possibility that such information might not be so easily available because of privacy reasons, etc.).

In this paper we present 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. Next, as a case study, we present an analysis of several of the most popular OMN routing algorithms in the literature. We implemented and evaluated them using the CCPAC simulator, and the obtained results offer a detailed view and a strong comparison between these algorithms. We present an analysis of their benefits and disadvantages, and an analysis of their performance on multiple traces.

The rest of the paper is structured as follows. We first present work related to our own. Section 3 presents details of the OMN routing algorithms under study. Sections 4-6 present the simulator, the mobility traces being used, and the analysis of the obtained results. Finally, Section 7 concludes the paper.

2. Related work

There have been several attempts to build simulation platforms or frameworks for delay tolerant networks [1-5]. The framework presented in [1] is based on OMNet++ [6], and it concentrates on providing simulated mobility patterns on which the algorithms are to be simulated. This simulated mobility patterns can easily be used as trace inputs for our simulator, as we proved with the Random Waypoint trace we generated in a similar manner.

The ONE Simulator [2, 3] is constructed in such a way that it accepts a range of movement models and event generators that can easily be interchanged as well as support for external DTN routing simulator. A powerful feature of this Simulator is the visualization part; it offers visuals for both result graphs as well as maps of the mobility patterns. In ONE the developer has to define different nodes for movement, persistency in data storage, energy consumption, etc. In CCPAC the developer has more flexibility, as he can define his own simulated models and entities.

The simulators used in [4] and [5] were created for a specific range of traces and algorithms. They were specifically constructed to create the results required for those specific papers.

The need for a simplified simulator can be observed in works such as [25] where multiple opportunistic routing algorithms are compared using multiple simulators including specially constructed ones.

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

3. OMN routing algorithms

There are a number of OMN routing algorithms, each of them addressing certain issues in such a network and requiring different data for a proper functionality. In this section we present 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.

3.1. Epidemic

Epidemic [7] is a well-known delay tolerant network routing algorithm. It relies on flooding (some articles even call it Flood): every packet[2] is sent to all the neighbors that do not already have it, and all packets are stored forever. As such, Epidemic has the maximum possible delivery ratio: all packets will arrive to their destination and the shortest possible path will eventually be used. This is why we use Epidemic as an upper (maximum) boundary for both delivery ratio and total cost. The problem with Epidemic is that it requires infinite resources. All packets must be stored, and when the network grows even slightly, the storage space required on every device increases dramatically.

3.2. Wait

The Wait routing algorithm (or Direct Delivery) [8] is the complete opposite of Epidemic. Unlike Epidemic, which sends as many copies of the message as possible, Wait only makes one transfer per message: directly to the destination. Encountering the destination for a message directly can prove to be extremely rare, and as such the delivery ratio is very low. The advantage of this algorithm is that it has a total cost equal to the delivery ratio and both of them are extremely low.

Because of the really small delivery ratio, this is also not an algorithm suited for real life scenarios; but just like Epidemic, it offers a very good comparison metric. Wait provides a lower bound for all the tests. Epidemic and Wait not only represent lower and upper bounds, but they also are the first confirmation that an algorithm works correctly as none should pass these limits.

[2] We refer here to *packet* as a generic term; it can represent an entire message being transferred (as in some papers), or a small chunk as part of a message (CCPAC includes both data units).

3.3. MCP

Multiple-Copy-multiple-hoP (or MCP) [9] is a routing algorithm very similar to Epidemic. Unlike Epidemic, the protocol that sends an unlimited number of copies for an unlimited number of hops, MCP limits these two variables. A version of MCP that has been proved to bring good results in the case of DTNs is the 4-copy-4-hop version [9].

MCP can be applied in real life scenarios and gives decent results in both delivery ratio and total cost; as such it is a good comparison for other algorithms.

3.4. dLife

dLife [10] is an opportunistic routing protocol that focuses on the dynamical aspect of social interactions between users part of the OMN. It makes use of weighted contact graphs as social structures that evolve in time. More specifically, the day is split in different time slices, and for each time slice a contact graph is computed; the evolution of a contact graph is computed daily.

The forwarding strategy is based on two measures: the social weight and the importance of a node, all of them computed by every node at the end of each time slice. The social weight represents the social strength between two nodes. It is based on the contact graph of a node and determined by taking all the contact graphs into consideration, with the one in the current time slice having the greatest contribution. The importance of a node is a measure that represents the social importance it has in the network viewed at a global level. It is calculated based on the node's degree in the contact graph and the social strength towards its neighbors. Thus, a packet is replicated in the network when the encountered node has a better relationship with the destination node, represented by a bigger social weight, or it has a greater importance than the current node.

3.5. Rank

The Rank algorithm [9] records the popularity of a node in time. The more connections a node has and the longer they hold the greater its rank increases. Messages are forwarded only to nodes that have higher ranks or to the destination. The algorithm stipulates that a popular node that has a lot of connections is more likely to meet the destination.

This algorithm is very simple to implement compared to others, but considering each node has to keep track of its own popularity and its own rank, a real world solution can prove to be more difficult as security issues need to be considered so that a node would not be able to exploit the network. There are also issues with how this popularity changes over time but this can be treated depending on the use case.

3.5. Label

Label [5] is a simple algorithm, very similar to Rank. Unlike Rank, Label presumes every node is part of one or multiple communities. Messages are forwarded to nodes that are in the same community with the destination node. The algorithm stipulates that nodes in the same community have higher chances of meeting each other and as such the messages would have a higher chance to reach its destination.

Label requires additional data about the nodes, the community they are in. There have been proposed a number of ways to obtain this data: social networks can offer a graph of friendships, interconnections between individuals and from this graph communities can be extracted; past network data can be exploited with a k-Clique algorithm to extract communities; there is also a distributed version proposed that solves this problem dynamically. Because this algorithm is extremely dependent on community data the results can vary dramatically with the method used to obtain this data. Furthermore applying this algorithm in real life scenarios can prove difficult as the data required may not always be available.

In this paper we do not present Label in more details, as we believe this algorithm is far too dependent on the community detection method and we have found no proof of a possible best alternative for all scenarios.

3.6. BUBBLE Rap A and B

These algorithms are among the most popular ones used in OMNs. *Pan Hui et al* [9] propose them as a merger between the Label and Rank algorithm.

The BUBBLE Rap algorithm bases its forwarding decision not only on the community or only on the popularity of the node, known as physical and social aspects of a network, but both of these characteristics are used simultaneously. In the forwarding process the nodes try to get the message to either the community of the destination node or to a popular node. It has already been proven that both of these characteristics raise the chance of a message to be delivered so using both of them increases the chances even more.

The difference between BUBBLE Rap A and B is that in B a message is deleted as soon as it is delivered to the community of the destination node. This causes a significant drop in the total cost with only a small difference in delivery ratio between them. We have tested both of these algorithms on all our traces.

3.7. PRoPHET

The PRoPHET Routing algorithm [14] uses contact history to compute the probability of re-encountering a node, more specifically the destination node of a message. This metric is called *Delivery Predictability*, $P(a,b) \in [0,1]$, calculated by every node *a* for each possible destination *b*. The value of this metric is high for nodes that come into contact often with *b* and decreases over time to minimize the effect of old encounters. Moreover, transitivity is introduced as a method to spread probability to nodes that do not encounter *b* directly but do so through an intermediary node. Thus, whenever two nodes encounter each other they exchange their calculated probabilities and each of them evaluate the probability of indirectly encountering the destination node.

The forwarding strategy follows this pattern: if an encountered node has a greater probability of delivering the message, then the message is forwarded to it. As such, the chance that a message reaches the destination increases with every hop.

3.8. PROPICMAN and CiPRO

The PROPICMAN Routing algorithm [15] uses context information to determine the path that maximizes the delivery probability in an intermittently connected mobile ad-hoc network. In the routing process only information about the destination node is used. Thus, each node constructs a node profile for the destination node that contains data such as device holder information (name, workplace, hobbies, etc.). This data is called evidence that a node has about the destination. It is assumed that the sender of a message already has a node profile of the destination. Each data entry in the node profile is weighted to reflect its importance and composes the header of the message.

The forwarding strategy of the PROPICMAN algorithm consists of sending the message on the two-hop route that has the highest delivery probability. This metric is computed by using the information in the message header and the information that an encountered node has about the destination.

The difficulty of generating relevant evidence for our synthetic traces, privacy issues or simple data unavailability prevented us from obtaining the traces needed for testing this algorithm. Thus, PROPICMAN is not implemented in the proposed simulation platform. Similar issues apply to CiPRO [16], a routing algorithm that brings improvements to PROPICMAN.

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 [12] or to use Gaussian processes to model encounters [13] 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 [12, 13, 17]. It does not only offer a way to simulate OMNs (although this is now its main purpose) but as shown in [17], 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 modules 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 [12] 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.

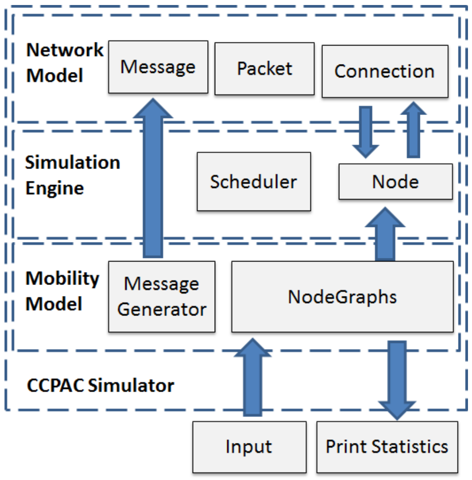

Fig. 1. Architecture of the simulator.

The modules that make the CCPAC simulator are:

- ***Connection*** – handles all the data pertaining to a connection, usually the 2 nodes that are part of the connection along with the start and end time of their encounter. The type or stability of a connection would be included here.
- ***Input*** – reads the trace inputs and parses them in the Node/Connection, the result should always be a NodeGraph. If it is preferred Input can be used to generate a trace. The simulator is built to function with mobility traces that offers a list of 2 or more device detections with timestamps.
- ***NodeGraph*** – Should not be changed, it stores the entire data required to run the simulation and it handles all the message transfers, this is the Core of the application.
- ***Message*** – Represents a message and all the data a message may contain. A message can be broken in multiple packets. The size of a message represents the number of Packets it needs to split into. In the simulation experiments presented here, we used a message size of 1. Larger message sizes can be used to analyze partial data transfers.
- ***MessageGenerator*** – Generates all the messages in the network, it is called at every simulation step and it makes the decision if more messages should be generated or not. For the presented results we use the MessageGeneratorExtended which splits the simulation time in 20 slices and generates 2 messages per node at each slice with a random destination from all the other nodes. The total number of messages equals the number of slices times the number of nodes times the number of messages per node.
- ***Node*** – represents the behavior of a single device in the network, there is one Node Object used for every device. The Node needs to implement its own initialize(), addPacket() and removePacket() as well as prepareSendPackets(). The last function should iterate over the Packet list and the list of current neighbors and call the NodeGraph every time it decides to make a transfer. For example in the case of Epidemic, which does not require for packets to ever be removed we have overwritten removePacket to do nothing. All the limitations should be handled in this module so that different algorithms or simulations can easily be tested with their own set of limits.
- ***Packet*** – represents one part of the message. To make the simulation more efficient only one copy of the Packet object exists at once in the network and only the reference to it changes. This can be changed like we did in the case of MCP where we built an object that extends Packet and goes

between it and the actual Packet, this way the ExtendedPacket could hold data such as number of Hops that each individual Packet has made.

- ***PrintStatistics*** – can be used to print any statistics that are needed, either from the Node or from the Messages.
- ***Scheduler*** – The scheduler is used to break a Message into Packets and place them in the Nodes at the time they are allowed to be transferred. By adding the message only then, we simulate a schedule. It was used in [17] to schedule Virtual Machine Transfers between physical machines inside hybrid clouds.
- ***SimulationTime*** – handles the passing of time in the simulation. We implemented 2 types of time passage: real time, where each second is processed and event based time where we only process events such as 2 devices starting a connection. SimulationTime controls the entire simulation, it fits behind the entire architecture.

We use a Factory to generate most of the modules presented, for instance if the Message type or packet type can be chosen from the configuration file and the Factory will generate the correct type for the entire simulation.

These modules work together 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 NG, the Node Graph and each Node, each device executes its own code. After one execution step the simulation time is recalculated and the next step is started.

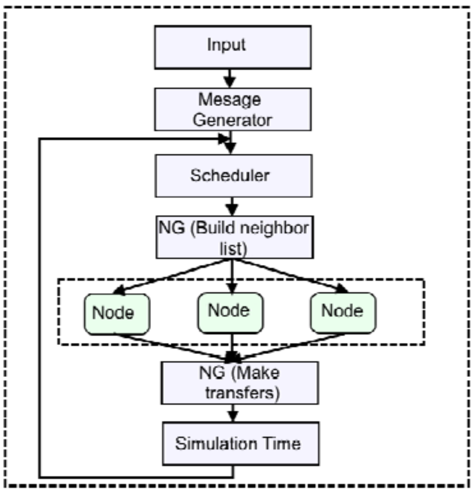

Fig. 2. Simulator workflow

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 quite simple. Figure 3 presents most of the code for the Wait algorithm, while Figure 4 presents most of the code the Epidemic algorithm. The difficulty of writing the code can vary from algorithm to algorithm but because of the way the simulator is built there is little effort needed to integrate it.

```
for(Node node : neighborNodes) {
   for(Packet p : packets) {
       if(p.message.destinationNode.equals(node)) {
           nodeGraph.addTransfer(p, this, node);
       }
   }
}
```

Fig. 1 Wait routing algorithm.

```
for(Node node : neighborNodes) {
   for(Packet p : packets) {
       if(!node.packets.contains(p)) {
           nodeGraph.addTransfer(p, this, node);
       }
   }
}
```

Fig. 2 Epidemic routing algorithm.

5. Traces used in our experiments

For our comparison of OMN algorithms we used five different real-world, publically available, mobility data traces, and one synthetic one. 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* [18] 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.

– In *standrews-sassy* [20] 25 mobile devices were distributed among participants from the University of St Andrews, composed of 22 undergraduate students, 3 postgraduate students, and 2 members of staff. The study lasted for over 2 months, starting on February 15th and until April 29th 2008.
– The *Infocom05* [21] trace includes Bluetooth sightings by groups of users carrying small devices (iMotes) for three days during the Conference IEEE Infocom in Grand Hyatt Miami. 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 [1]. 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 1 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 [12,13,19].

Table 1. Characteristics of the Experimental Data Sets

Experimental data set	**Duration (days)**	**Number of Internal Experimental Devices**	**Number of Internal Contacts**
Reality	246	97	113,367
StAndrews	74	25	112,264
Infocom05	3	98	22,459
Random	30	30	2,871

6. Simulated results

The following results were obtained using the presented simulator. We chose to present the Reality, StAndrews, Infocom05 and Random waypoint traces as this set of traces cover most of the possible scenarios.

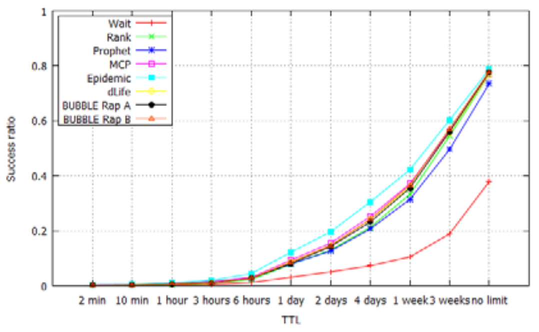

Fig. 3 Reality Delivery Ratio

Furthermore we offer the reality trace to support a direct comparison with the work conducted by *Pan Hui et al* [9]. This proves the validity of our implementation.

As seen in Figure 5 to 12, we offer the results in the two metrics that are most relevant to the working of an OMN routing algorithm. These metrics are the *Delivery Ratio* and the *Total Cost*. We chose to not add any routing table information because of lack of space and because most of the presented algorithms do not use a routing table at all.

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

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

- The *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:

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

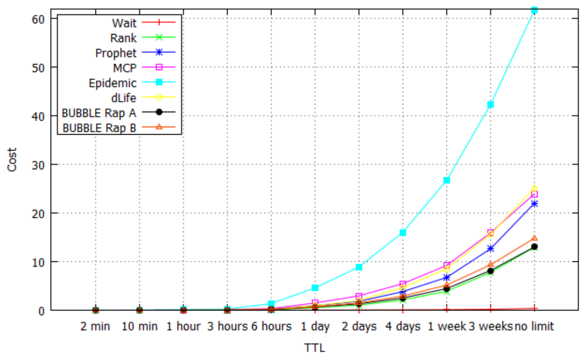

Fig. 4 Reality Total Cost

In case of Reality (Figures 5 and 6), we observe a high delivery ratio for most algorithms. Even the Wait algorithm has a delivery ratio of 0.4. This happens because the Reality trace spans over a very long period of time, almost a year, and most nodes will eventually meet most of all other nodes.

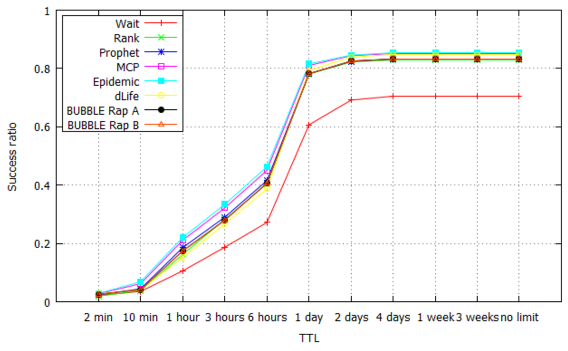

Fig. 5 Infocom05 Delivery Ratio

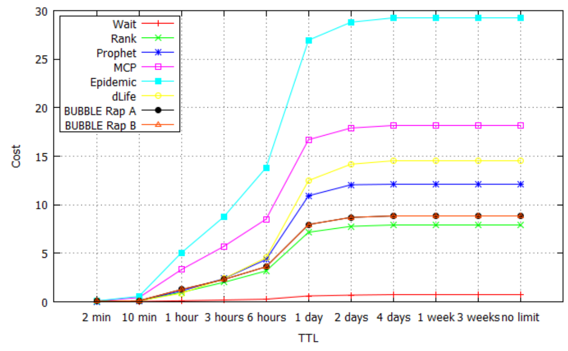

Fig. 6 Infocom 05 Total Cost

Unlike the Reality trace which spans over a year, the Infocom05 trace (Figures 7 and 8), 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 presented algorithms, with the exception of Wait, costs are clearly distinct. 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.

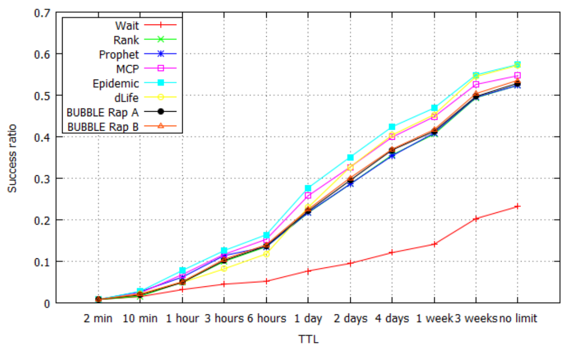

Fig. 7 StAndrews Delivery Ratio

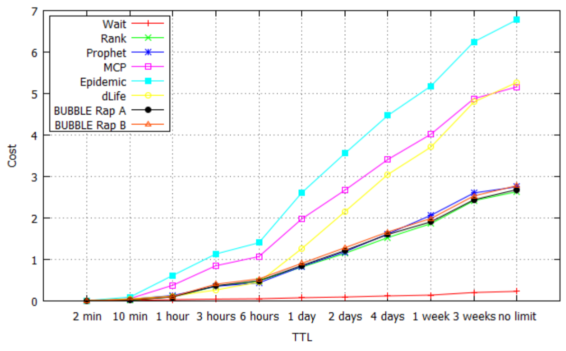

Fig. 8 StAndrews Total Cost

StAndrews (Figures 9 and 10), is a trace that differs from both Infocom and Reality. With the long duration of 2 months and high number of contacts, StAndrews represents an ideal scenario. In this trace we can observe clear differences between the algorithms in both total cost and delivery ratio but 4 algorithms manage to bring extremely similar results. These algorithms are: Prophet, BUBBLE Rap A and B and Rank. The results vary greatly with every trace. But what we did not expect to vary as much was the overall performance of the algorithms compared to each other. For instance Prophet performs extremely well in the StAndrews trace, but it does not offer such performance in the other traces we performed.

Random waypoint (Figures 11 and 12), is a synthetic trace we constructed to obtain a better comparison of the performance of the presented algorithms. The results are very similar with the ones obtained from Infocom 05, and this 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.

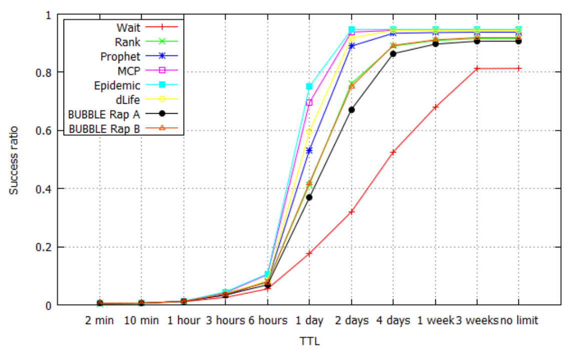

Fig. 9 Random Waypoint Delivery Ratio

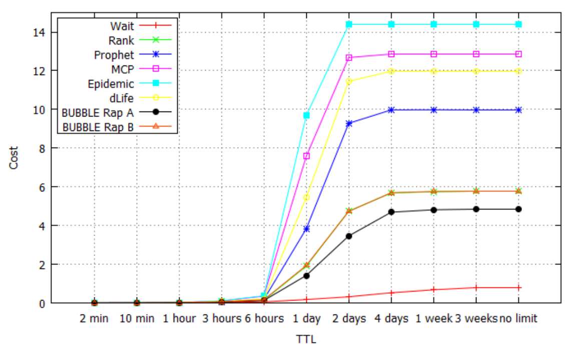

Fig. 10 Random Waypoint Total Cost

Figure 13 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.

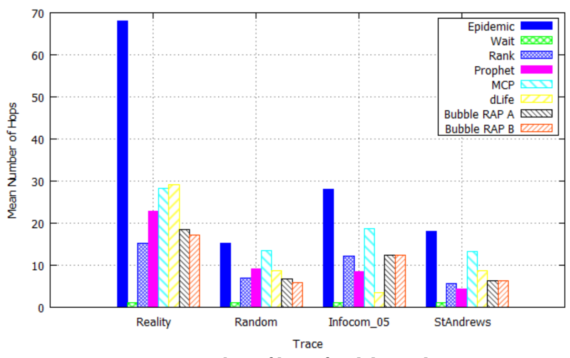

Fig. 13 Mean Number of hops for delivered messages

In comparison to Figure 13, in Figure 14 we show latency in time as a mean over all delivered messages. Because the time intervals of the traces differs 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.

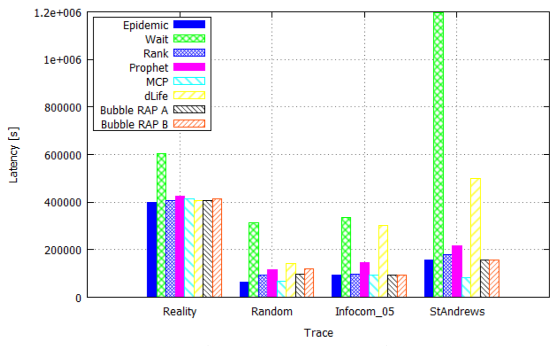

Fig. 14 Latency of delivered messages for all algorithms

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. Conclusions

In this paper we presented 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.

Acknowledgments

The research presented in this paper is supported by national project MobiWay, Project PN-II-PT-PCCA-2013-4-0321, no. 16/2014.

References

[1] A. Petz, J. Enderle, C. Julien. *A Framework for Evaluating DTN Mobility Models*. In Proc. of the 2nd International Conference on Simulation Tools and Techniques (Simutools '09). Brussels, Belgium, Article 94, 8 pages, 2009.

[2] A. Keranen, J. Ott. *Increasing Reality for DTN Protocol Simulations*. Tech. rep., Helsinki University of Technology, Networking Laboratory, July 2007.

[3] A. Keranen, J. Ott, T. Karkkainen. *The ONE Simulator for DTN Protocol Evaluation*. In Proc. of the 2nd International Conference on Simulation Tools and Techniques (Simutools '09). Brussels, Belgium, Article 55, 10 pages, 2009.

[4] A. Islam, M. Waldvogel. *Reality-Check for DTN Routing Algorithms*. In Proc. of the 28th International Conference on Distributed Computing Systems Workshops (ICDCSW '08). Washington, DC, USA, pp. 204-209, 2008.

[5] P. Hui, J. Crowcroft. *How Small Labels Create Big Improvements*. In Proc. of the 5th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW '07). Washington, DC, USA, pp. 65-70, 2007.

[6] A. Varga, R. Hornig. *An overview of the OMNeT++ simulation environment*. In Proc. of the 1st international conf. on Simulation tools and techniques for communications, networks and systems & workshops (Simutools '08). Brussels, Belgium, Article 60, 10 pages, 2008.

[7] A. Vahdat, D. Becker. *Epidemic routing for partially connected ad hoc networks*. Technical Report CS-200006, Duke University, 2000.

[8] T. Spyropoulos, K. Psounis, C.S. Raghavendra. *Single-copy routing in intermittently connected mobile networks*. In Proc. Of First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON 2004), pp.235-244, Oct. 2004.

[9] P. Hui, J. Crowcroft. *BUBBLE Rap: Social-Based Forwarding in Delay-Tolerant Networks*. IEEE Transactions on Mobile Computing, vol. 10, no.11, pp. 1576–1589, 2011.

[10] W. Moreira, P. Mendes, S. Sargento. *Opportunistic routing based on daily routines*. In Proc. of IEEE Int. Symp. World of Wireless, Mobile and Multimedia Networks (WoWMoM 2012), pp. 1-6, June 2012.

[11] F. Bjurefors, P. Gunningberg, C. Rohner, S. Tavakoli. *Congestion avoidance in a data-centric opportunistic network*. In Proceedings of the ACM SIGCOMM workshop on Information-centric networking (ICN '11). New York, NY, USA, pp. 32-37, 2011.

[12] C. Chilipirea, A.-C. Petre, C. Dobre. *Energy-Aware Social-based Routing in Opportunistic Networks*. In Proc. of the 27th International Conference on Advanced Information Networking and Applications Workshops (WAINA 2013), Barcelona, Spain, pp. 791-796, 2013.

[13] C. Chilipirea, A.-C. Petre, C. Dobre. *Predicting Encounters in Opportunistic Networks using Gaussian Process*. In Proc. of 19th int. conf. on Control Systems and Computer Science (CSCS19), Bucharest, Romania, 2013.

[14] A. Lindgren, A. Doria, O. Schelen. *Probabilistic Routing in Intermittently Connected Networks*. SIGMOBILE Mob. Comput. Commun. Rev. 7(3), pp. 19-20, July 2003.

[15] H. Anh Nguyen, S. Giordano, A. Puiatti. *Probabilistic Routing Protocol for Intermittently Connected Mobile Ad hoc Network (PROPICMAN)*. In Proc. of IEEE Int. Symp. on World of Wireless, Mobile and Multimedia Networks (WoWMoM 2007), pp. 1-6, June 2007.

[16] H. Anh Nguyen, S. Giordano. *Context information prediction for social-based routing in opportunistic networks*. Ad Hoc Networks, Vol. 10, Issue 8, pp. 1557-1569, Nov. 2012.

[17] M.-C. Nita, C. Chilipirea, C. Dobre, F. Pop. *A SLA-Based Method for Big-Data Transfers with Multi-Criteria Optimization Constraints for IaaS*. In Proc. of the 12th RoEduNet Conference: Networking in Education and Research, Constanta, 2013.

[18] N. Eagle, A. Pentland. *Reality Mining: Sensing Complex Social Systems*. Personal and Ubiquitous Computing, vol. 10, no. 4, pp. 255-268, May 2006.

[19] R.-I. Ciobanu, C. Dobre. *Social Aspects to Support Opportunistic Networks in an Academic Environment*. In Proc. of the 11th int. conf. on Ad-hoc, Mobile, and Wireless Networks (ADHOC-NOW'12), Springer-Verlag Berlin Heidelberg 2012, LNCS 7363, pp. 69–82, 2012.

[20] L. Pelusi, A. Passarella, M. Conti. *Beyond MANETs: dissertation on Opportunistic Networking*. IITCNR Tech. Rep., May 2006.

[21] J. Scott, R. Gass, J. Crowcroft, P. Hui, C. Diot, A. Chaintreau, {CRAWDAD} trace cambridge/haggle/imote/infocom (v. 2006-01-31), Available since Jan. 2006.

[22] J. Scott, R. Gass, J. Crowcroft, P. Hui, C. Diot, A. Chaintreau, {CRAWDAD} trace cambridge/haggle/imote/infocom2006 (v. 2009-05-29), Available since May 2009.

[23] L. Pelusi, A. Passarella, M. Conti. *Opportunistic networking: data forwarding in disconnected mobile ad hoc networks*. IEEE Communications Magazine, vol.44, no.11, pp.134-141, Nov. 2006.

[24] S. Jain, K. Fall, R. Patra. *Routing in a delay tolerant network*. SIGCOMM Comput. Commun. Rev. 34, 4 (August 2004), pp. 145-158, 2004.

[25] Bulut, Eyuphan. Opportunistic routing algorithms in delay tolerant networks. Diss. Rensselaer Polytechnic Institute, 2011.