Opportunistic Network Behavior in a Swarm: Passing Messages to Destination

Ahmad Reza Cheraghi, Julian Zenz, Kalman Graffi

Technology of Social Networks, Heinrich Heine University Düsseldorf, Germany
Honda Research Institute Europe GmbH, Offenbach am Main, Germany
ahmad.cheraghi@hhu.de, julian.zenz@hhu.de, kalman.graffi@honda-ri.de
https://tsn.hhu.de, https://www.honda-ri.de

***Abstract*—This paper evaluates and compares two routing algorithms for message passing in opportunistic peer-to-peer networks in the swarm network simulator Swarm-Sim. The two considered algorithms, Epidemic Routing and PRoPHET Routing, are compared in terms of delivery rates, efficiency and overhead, while presenting different mobility models. By this, we combine a variation of mobility models and a set of communication protocols, measuring their impact on the swarm behavior. Message delivery rates for Epidemic routing are considerably and consistently better. However, PRoPHET generates less overhead than Epidemic Routing independent of mobility models and scenarios. The measured overhead of Epidemic exceeded that of PRoPHET by about 100%. Memory requirements for keeping delivery probabilities - used by PRoPHET to circumvent flooding - might be a limitation as they must be kept locally per peer, this detail offers aspects for further research in the future. Finally, this paper demonstrates Swarm-Sim to be a viable alternative to simulate routing protocols in opportunistic peer to peer networks without requiring great insight into the simulator's architecture and thus establishing itself as an easy to use simulator. It achieves simplicity without yielding any of flexibility of application.**

I. Introduction

This paper evaluates two routing protocols for message passing in *opportunistic networks* (OppNets) by simulation and demonstrates implementation and evaluation simplicity of these two protocols in different scenarios using the swarm network simulator *Swarm-Sim*.

According to Cheraghi et al. OppNets are considered to be a subcategory of delay-tolerant networks (DTN), which are networks – in terms of message passing – where a link between the sender and receiver of a message may never be established, neither directly nor indirectly via different nodes [1]. Moreover, Jain et al. note that in order to make message passing in a DTN actually feasible, all nodes have to apply the store-and-forward principle, i.e. each node has to be a potential messenger for every message inside the network [2].

An example of an OppNet is a network of people moving inside cities or urban areas. In a scenario of catastrophe all cellular networks are down but text messaging via their smartphones is still possible. Each mobile person carries a smartphone, which is equipped with transmission chips for Bluetooth or WiFi for close range communication and message transfer. However, there are may be no direct links between sender and receiver of a message in such networks. Message passing in such a scenario can be problematic, since the mobile sender and receiver may never cross paths or the sender may not be within distance of a potential messenger for a long time. This is were routing algorithms for OppNet attempt to provide communication in unstable and unreliable conditions.

Another, similar scenario may be the movement of drones that scan an area for a missing person. The network nodes communicate with each other and pass along messages about findings to a remote station. Due to their size, the drones are only equipped with antennas for close range communication and cannot pass message directly to the remote station and must therefore rely on opportunistic networking between nodes to extend their range of communication.

Considering the limitations within such scenarios and the characteristics of OppNets, several routing protocols have been developed to work in such environments. The two algorithms examined in this paper are epidemic routing[3] and PRoPHET routing [4]. These two algorithms were chosen because they take almost diametric approaches. While epidemic routing is based upon the premise of flooding the network by taking every chance of replicating a message, nodes in PRoPHET behave more moderately and carefully calculate delivery probabilities and only replicate messages when the algorithm deems this to increase delivery probability. These different principles should result in equally opposing outcomes in terms of metrics such as delivery probability of a single message and efficiency, i.e. how many replications did it take for a single message to be delivered.

Before any such routing protocol can be tested in the real world, it is very common to simulate scenarios and fine tune parameters and settings. This is performed in Swarm-Sim. Other simulators do not consider simulation of swarms as a main focus. Swarm-Sim addresses these shortcomings while maintaining its flexibility and ease of use, as will be demonstrated by the evaluation of OppNet routing in this paper.

In a nutshell, this paper aims to demonstrate the capability of Swarm-Sim to evaluate routing protocols in OppNets and in essence provide arguments to underscore the simulator as a viable alternative to previously published simulators.

The paper will be structured as follows: first related simulators as well as Swarm-Sim will be presented in further detail within the following section II. This is succeeded by an brief overview on Epidemic and PRoPHET routing and theoretical background on the implementation specifics of

message passing in Swarm-Sim (III). Then the metrics used to evaluate those protocols and the experimental setup are explained in the methodology section (IV). The second to last section discusses the results (V), leading to the conclusion (VI).

II. Related work

This second section provides a small overview of two OppNet simulators related to Swarm-Sim, which was used in this project. The routing protocols PRoPHET and Epidemic will be explained in more detail in section III.

A. ONE Simulator

Keranen et al. describe *Opportunistic Network Environment Simulator* (ONE) as one of the first tools to evaluate OppNets [5]. The main aspects of ONE are simulating the movement of nodes, communication between nodes and routing as well as processing messages [5]. In summary, ONE is an environment to test routing in opportunistic networks. One may use the provided algorithms or implement their own. In terms of architecture, ONE is divided into separate modules. Each of which takes care of different task. Routing is one of these modules, the others describe the movement of nodes, the visualization or the log events inside the simulator. Movement of nodes is round-based and depends heavily on its capabilities, such as energy capacity.

B. DTN Agent

DTN Agent [6] "is implemented as add-on module for the ns-2 simulator to deploy DTN behavior over the traditional TCP/IP stack" and focuses on simulating "space internetworking" [1]. As of 2012, it is no longer being maintained.

C. Swarm-Sim

Swarm-Sim was developed by the department of social networks at Heinrich-Heine-University Düsseldorf. It is a 2D and 3D simulation core that provides interfaces for creating individual scenarios and solutions with the main focus of simulating swarms. This simulator is open source and can be downloaded from [7]. The cooperation and interactions between the robots and their environment are both modeled by the simulator. It thus describes the modeling of operations that a robot performs in its environment and the connections between all the components involved in the same system It is a round-based simulator that performs entirely configurable solutions inside a user-created world of different types of matter. The simulator consists of a *configuration file*, a *scenario module*, a *solution module*, a *visualizing module* and a *CSV-generator*. Figure 1 shows the architecture of swarm-sim.

Simulations in Swarm-Sim are configured via a configuration file (`config.ini`). Users can change different parameters outside the actual code via this file. The environment is a virtual 2D space where robots and other matters are placed on a hexagonal space (c.f. figure 2) and tasks are performed. The types of matter that can be placed into an environment include *tiles*, *particles* and *locations*. Particles represent the autonomous agents that can move themselves, tiles are items that cannot move but can be transported by the particles. Lastly,

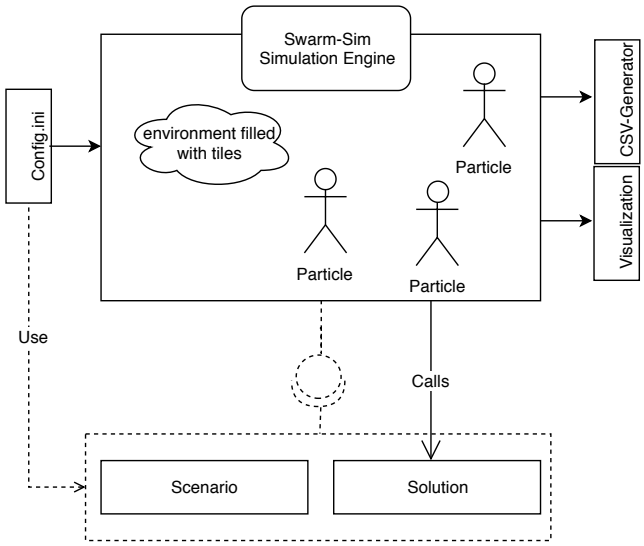


Fig. 1: Swarm-Sim architecture.

locations are the coordinates in an environment where the tiles and particles can be placed. Tiles can act as obstacles or objects, that can be transported by the particles. All three types of matter have a memory to store information that can be accessed and manipulated by other particles.
The environments constructed by the users are stored in the *Scenario* module. Swarm-Sim has a csv generator (`csv_generator.py`) to produce csv files that include the results of the simulations. This file collects all the statistical data of particles that is generated during the simulations. For example, the number of tiles created or deleted by a particle, number of tiles or particles read or written by a particle. Along with that, the different parameters used in the simulations are also collected. For example, maximum number of rounds and number of movements needed for a specific task. The collection of data is done at the end of each simulation. This makes it possible to evaluate and compare results of each simulation. The results are shown in a separated file, namely under outputs. This module was easily extended in order to count metrics for passed messages inside the network.

Moreover, to see every simulation process as an animation, Swarm-Sim has a visualization module known as `vis.py`, which is responsible for presenting the simulations and taking screenshots. It displays the simulation-process graphically in a separate window. This gives users the opportunity to examine the particle behaviors when accomplishing different tasks.

In order to simulate a routing protocol like Epidemic Routing or PRoPHET in Swarm-Sim, the *Solution* module was used. It defines the steps or algorithm the simulator runs. In this case it would define the behavior of particles within the two routing protocols.

With regards to message passing between particles in Swarm-Sim, the particle's ability to read and write to another particle's memory was used. This allows to build a network with neighborhood-communication and evaluate routing algorithms with the simulator's built-in tools. A link to the real

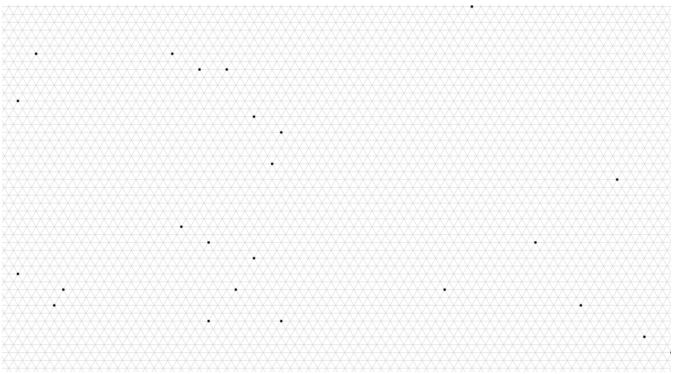


Fig. 2: Excerpt of a 200 by 200 world in Swarm-Sim illustrating the hexagonal structure. The black dots represent particles.

world could be to interpret the particles as smartphones or drones, which have a limited memory to store-and-forward messages and travel with their users through a world, scanning for other nodes to pass their messages onto. Within Swarm-Sim this is performed by scanning the surroundings for other nodes. This procedure will be explained in more detail in subsection III-A.

III. Routing protocols and theoretical background

The following section gives a brief insight into the theoretical background of Swarm-Sim and also explains the examined routing protocols Epidemic and PRoPHET routing in more detail.

A. Theoretical background on Swarm-Sim

Let P be the set of all particles in a Swarm-Sim simulation and (max_x, max_y) be the maximum width and height of the Swarm-Sim world. The former is configured inside the scenario and the latter is part of the config (cf. figure 1). Then each particle $p_i \in P$ has a position vector $q_i = (x_i, y_i) \in \mathbb{L}$, with

$$\mathbb{L} := \{(x, y) \mid x \in \mathbb{Z}_{\pm 0.5}, y \in \mathbb{Z}, |x| < max_x \wedge |y| < max_y\}, \tag{1}$$

where:

$$\mathbb{Z}_{\pm 0.5} := \mathbb{Z} \cup \{x \pm 0.5 \mid x \in \mathbb{Z}\}$$

$\mathbb{L}$ is called the set of *locations* and restricts the possible positions of particles due to the hexagonal structure of Swarm-Sim. The neighboring set of locations around a position vector $q_i = (x_i, y_i)$ is then defined as:

$$\begin{aligned}\mathbb{L}_i := \{(x, y) \mid (x, y) \in \mathbb{L}, &(x = x_i \pm 1 \wedge y = y_i) \vee \\ &(x = x_i \pm 0.5 \wedge y = y_i \pm 1)\},\end{aligned} \tag{2}$$

where $|\mathbb{L}_i| = 6$. Furthermore, we define the injective function $Q : P \to \mathbb{L}$, which maps distinct matter onto its location, i.e. for each particle $p_i \in P$:

$$Q(p_i) := q_i,$$

where q_i is the location of particle p_i.

The particles in Swarm-Sim have a limited memory, which consists of a dictionary object and which size can be adjusted using the configuration. In our simulation, each particle creates a dictionary object in the particle's memory with the unique identifier of the receiver of a message as the key. This key-value storage contains the message objects, each referencing the sender, receiver and including the message content. It uses the unique identifier of a message as key.

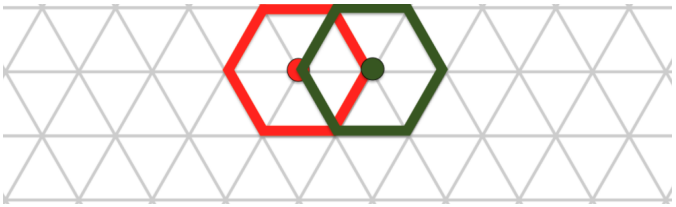


Fig. 3: Two particles (red and green dot) in close proximity and their communication range (red and green hexagon respectively).

We limit communication between particles to their neighbors. Figure 3 shows two particles in close proximity with overlapping communication range. If we assume the set of all particles in a simulation P to be of size n, i.e. $|P| = n$, then let p_i denote the i-th particle. The set of particles with which p_i can interact, depends on its interaction range ω_i and is defined as

$$N_i := \{p_j \mid p_j \in P, d(q_i, q_j) \leq \omega_i\}, \tag{3}$$

where $d(q_i, q_j)$ is a distance function for two locations, such that:

$$d(q_i, q_j) = \begin{cases} |x_j - x_i|, \text{if } y_i = y_j \wedge x_i \neq x_j \\ |y_j - y_i| + |x_j - x_i| - |y_j - y_i| * 0.5, \\ \text{if } |x_j - x_i| - |y_j - y_i| * 0.5 > 0 \\ |y_j - y_i|, \text{else.} \end{cases} \tag{4}$$

Let M be the set of all messages in a simulation. When one particle $p_i \in P$ (e.g. the green one in figure 3 encounters another particle $p_j \in P$ (e.g. the red one in figure 3)) and decides to transmit messages to p_j, then p_i loops through its memory, which is keyed by receiver identifiers, and checks whether p_j already has any of the messages $m \in M$ for a receiving particle $p_r \in P$. If p_j does not have one of the messages m for the receiving particle p_r, then the green particle p_i writes m to the memory of the red particle p_j. If p_i finds that $p_j = p_r$, then it has found the receiver of a message m and conclusively deletes m from its own memory.

This process is the same for both Epidemic and PRoPHET routing. The differences between the protocol are in essence realized in the decision of transmitting messages from a particle p_i to an encountered particle p_j for a specific receiver p_r. While in Epidemic routing p_i will transmit every message it has for any receiver p_r, p_i will decide the message transmission for every p_r depending on the previously explained delivery probabilities but will then also transfer every message it has for the specific receiver p_r.

B. Epidemic Routing

Epidemic Routing builds on the premise of flooding inside an OppNet. The principle of this routing algorithm is simple

and easy to implement: each particle in Swarm-Sim scans for other particles in their vicinity and forwards all messages it has to the discovered particles. If by chance the destination is reached, the particle removes this message from its memory, otherwise not. Therefore, the protocol implementation allows duplicate deliveries of a single message. The only way to tweak this routing protocol is to limit the scan radius of each particle.

C. PRoPHET Routing

PRoPHET routing aims to improve upon the flooding nature of Epidemic Routing by considering delivery probabilities. In our implementation each particle p_i stores an estimated delivery probability $\mathbb{P}_\epsilon(p_i, p_j)$ and the time of the last encounter for all other particles $p_j \in P$. $\mathbb{P}_\epsilon(p_i, p_j)$ is the probability that p_i can deliver a message to p_j. This implies that particles must be aware of all other particles in their world. Each delivery probability is initialized to a specific constant $\mathbb{P}_{init}$ (default: .75) and the time of the last encounter t_e is simply the simulator round. The probability $\mathbb{P}_\epsilon$ changes depending on whether or not the particle p_j is the receiver of a message that p_i carries, the time of the last encounter and delivery probabilities of other particles that p_i encountered.

More specifically, $\mathbb{P}_\epsilon(p_i, p_r)$ ages every round for each receiving particle p_r that p_i has a message for, using the following formula:

$$\mathbb{P}_\epsilon(p_i, p_r) = \mathbb{P}_\epsilon(p_i, p_r) * \gamma^{t_c - t_e}, \tag{5}$$

where t_c is the current and t_e is the last simulation round that p_i encountered p_r, which will be 0 if p_i has not encountered p_r yet.

Furthermore, $\mathbb{P}_\epsilon(p_i, p_j)$ changes for each particle p_j that p_i encounters by terms of

$$\mathbb{P}_\epsilon(p_i, p_j) = \mathbb{P}_\epsilon(p_i, p_j) + (1 - \mathbb{P}_\epsilon(p_i, p_j)) * l_e, \tag{6}$$

where l_e is a constant (default: .75) that scales the impact of encountering another particle.

Finally, if the particle p_i encounters a particle p_j it will update its delivery probabilities for each receiving particle p_r according to those of p_j as follows

$$\begin{aligned}\mathbb{P}_\epsilon(p_i, p_r) = \mathbb{P}_\epsilon(p_i, p_r) + (1 - \mathbb{P}_\epsilon(p_i, p_r)) * \\ \mathbb{P}_\epsilon(p_i, p_j) * \mathbb{P}_\epsilon(p_j, p_r) * \beta,\end{aligned} \tag{7}$$

where β is a constant (default .99).

Before a particle p_i updates its delivery probability for a receiver p_r using equation 7, it tests whether the probability $\mathbb{P}_\epsilon(p_j, p_r)$ is equal or greater than its own probability for delivering a message to the receiver, i.e. is $\mathbb{P}_\epsilon(p_j, p_r) \geq \mathbb{P}_\epsilon(p_i, p_r)$ or not. If this is the case, p_i forwards all of its messages for p_r to p_j (without producing duplicates). The principle idea behind this is that p_i only forwards a message if the delivery probability for that specific message via the encounter p_j, is at least equally as high as delivering the message itself in order to reduce flooding.

Previous research has found PRoPHET to have less communication overhead and send less messages, which seems natural due to the flooding principle of Epidemic Routing. The delivery rate of PRoPHET was also found to be comparable to that of Epidemic Routing, at least in scenarios where the nodes move randomly [4].

TABLE I: Particle parameters used in the experimental setup.

Parameter name	Parameter values
Mobility model structure	{random-walk, random-circles}
Mobility model steps	[10, 30]
Scan radius	{5, 10}

IV. Methodology

In order to compare and successfully infer conclusions about parameters of specific scenarios in OppNet routing, this following section will provide details of the experimental setup, as well as introduce the metrics used to evaluate these protocols in Swarm-Sim. The code is open-source and can be downloaded from [8].

A. Experimental setup

The experiments in this work were performed using several different configurations. Each configuration is defined by some parameters, which can be divided into two categories:

1) Particle parameters
2) World/scenario parameters.

The parameters of category 1 and their different configurations are shown in Table I. A *mobility model* defines the movement of a particle, i.e. the structure, which is either *random-walk* or around *random-circles*, (i.e. hexagons) and the steps, i.e. the length of movement without changing direction, which is restricted by an interval of minimum and maximum length. The length of this movement therefore defines how many steps a particle takes before it changes its direction either random or along the next edge of the hexagon.

The *scan radius* of a particle is the hexagon of the respective size in which a particle scans for other particles to transfer its messages to. Every particle within this hexagon has a hop distance (cf. equation 4) equal to the scan radius.

The mobility model called *random-circles* means that particles move along a hexagon of a random size, which changes every time the particle has completed that hexagon. The starting direction is also random. However, the direction along the hexagon is always counter-clockwise.

Additionally, the other mobility model is called *random-walk*. In this mobility model a particle moves in a random but fixed direction for a random number of steps in the range of [10, 30] (cp. *Mobility model steps* I) until it changes its direction to another random direction for a new random number of steps ranging from 10 to 30.

Table II shows the parameters used to configure the world and solution (category 2). The *initial particle placement* was always random but consistent across different particle parameters and routing protocols. The particle placements were generated beforehand by placing them randomly across the world.

The reasoning for this – as well as the two randomized mobility models – is to follow the idea of opportunistic networking as close as possible. Connections between sender and receiver should be established by chance and be unpredictable

TABLE II: World and solution parameters used in the experimental setup.

Parameter name	Parameter values
(max_x, max_y)	(200, 200)
Initial particle placement	random
Particle count	{200, 400}
New message interval	Every 5th round
Max round	1000

TABLE III: PRoPHET specific parameters used in the experimental setup.

Parameter name	Parameter values
$\mathbb{P}_{init}$	0.75
β	0.25
γ	0.99

instead of being due to the structure of the world or the movement of nodes.

The *new message interval* defines after how many rounds a new message is created and sent. Sender and receiver, as well as content, were also randomized. Considering the *max round* value was fixed to 1000, particles created 200 messages.[1]

Apart from these parameters, PRoPHET also uses several internal parameters for determining delivery probabilities, as was demonstrated in equations 5, 6 and 7 and their values are displayed in table III. The names and default values of these parameters were adopted from Lindgren et al. [4].

B. Metrics

Before we can define the metrics used to evaluate the experiments, we need to define a few sets. Let M be the set of all messages, then let M_D be the set of all delivered messages (without duplicates) and M_{ND} the set of all messages not delivered. Furthermore, let MS_D be the multi-set of messages that were delivered including duplicate deliveries. In addition, let MS_F be the multi-set of forwarded messages including all duplicates.

In order to measure delivery success, we define $\zeta := \frac{|M_D|}{|M|}$ as the delivery rate for the set M of messages. This may be the most important metric since any routing protocol that does not offer sufficient delivery success is simply not very useful.

As for efficiency per delivery we define $\eta := \frac{|M_D|}{|MS_D|}$ as the average delivery rate of duplicates, i.e. closer this value is to 1, the less duplicates were delivered inside the network. The delivery success rate alone is not enough to decide on practicality of a routing protocol in an OppNet since less efficiency results in more memory requirements per node, which can be very limited in the aforementioned example of drones or smartphones.

Finally, we define the real overhead for the message passing protocol as $\sigma := \frac{|MS_F|}{|M|}$, i.e. the average number of replicas per generated message or average number of hops per message.

[1]Although the simulator is stopped right after creating the last message, the message still has the chance to be delivered during this very last round.

TABLE IV: Delivery Success ζ for Epidemic and PRoPHET Routing.

Parameters	ζ Epidemic	ζ PRoPHET	Δ
{random-walk, 200, 5}	0.150	0.020	-0.050
{random-walk, 200, 10}	0.175	0.055	-0.120
{random-walk, 400, 5}	0.230	0.055	-0.175
{random-walk, 400, 10}	0.115	0.130	0.015
{random-walk, 600, 5}	0.385	0.065	-0.032
{random-walk, 600, 10}	0.085	0.135	0.050
{random-circle, 200, 5}	0.155	0.020	-0.135
{random-circle, 200, 10}	0.165	0.090	-0.075
{random-circle, 400, 5}	0.625	0.250	-0.375
{random-circle, 400, 10}	0.120	0.055	-0.065
{random-circle, 600, 5}	0.700	0.140	-0.560
{random-circle, 600, 10}	0.275	0.130	-0.145

V. Results and discussion

The previously described sets of parameters resulted in a total of of twelve experiments to which results we will turn to in the following.

A. Delivery Success

The delivery success rates ζ for each experiment are presented in table IV. The sets of parameters in the first column are sets which include the mobility model structure first, then the particle count, followed by the scan radius. Other parameters were set as explained in section IV.

Considering the Δ-values, it is immediately striking that Epidemic Routing seems far superior to PRoPHET in terms of ζ. Only twice, both with the random-walk mobility model and a scan radius of hops, are more of the 200 total messages delivered using PRoPHET than Epidemic for particle counts of 400 and 600, respectively. This is further underscored by the fact that the maximum observed ζ for PRoPHET is merely 25% but as high as 70% for Epidemic. In addition, seven of twelve ζ values for PRoPHET are lower than the minimum value 0.115 for Epidemic and were found to be as low as 2% for both random-walk and random-circle with 200 particles and a scan radius of five.

Overall, the ζ values seem to be rather unstable when comparing increasing particle count and the larger scan radius, especially for PRoPHET (cp. Figure 5). At least for Epidemic routing, a larger particle count seems to correlate with higher delivery rates (cp. Figure 4). Interestingly, ζ indeed seems to be negatively correlated to an increasing scan radius in several simulated instances of Epidemic. This is counter intuitive as higher scan radii should result in a higher probability of encountering other nodes and therefore increase and not reduce ζ. This requires closer inspection in the future as. Both protocols produce the highest delivery rates for the random-circle mobility model with a scan radius of five.

B. Delivery Efficiency

As was expected, delivery success rates of Epidemic exceeded those of PRoPHET, however, this is to the massive disadvantage of delivery efficiency η. As can be seen in table V, Epidemic produces high amounts of multiple deliveries per message.

When considering the low delivery rates for PRoPHET from the previous section, it seems almost obvious that η must

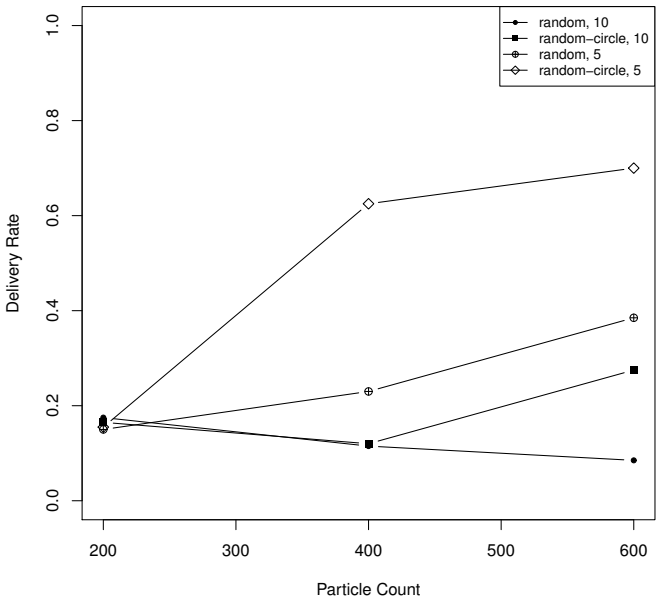


Fig. 4: Delivery Success ζ of Epidemic as a function of particle count for the different mobility models and scan radii.

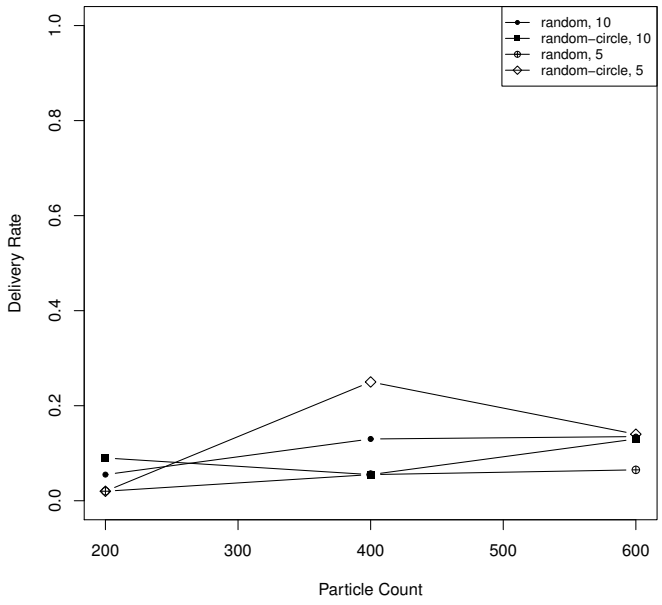


Fig. 5: Delivery Success ζ of PRoPHET as a function of particle count for the different mobility models and scan radii.

TABLE V: Delivery Efficiency η for Epidemic and PRoPHET Routing.

Parameters	η Epidemic	η PRoPHET
{random-walk, 200, 5}	0.395	1.000
{random-walk, 200, 10}	0.124	1.000
{random-walk, 400, 5}	0.619	1.000
{random-walk, 400, 10}	0.319	1.000
{random-walk, 600, 5}	0.526	1.000
{random-walk, 600, 10}	0.230	0.400
{random-circle, 200, 5}	0.155	1.000
{random-circle, 200, 10}	0.686	1.000
{random-circle, 400, 5}	0.351	1.000
{random-circle, 400, 10}	0.120	0.491
{random-circle, 600, 5}	0.391	0.867
{random-circle, 600, 10}	0.037	0.403

TABLE VI: Overhead σ for Epidemic and PRoPHET Routing.

Parameters	σ Epidemic	σ PRoPHET
{random-walk, 200, 5}	318.565	112.265
{random-walk, 200, 10}	1225.945	510.83
{random-walk, 400, 5}	1324.385	479.035
{random-walk, 400, 10}	4783.535	1955.125
{random-walk, 600, 5}	2979.19	1122.46
{random-walk, 600, 10}	10341.215	4827.07
{random-circle, 200, 5}	344.58	112.04
{random-circle, 200, 10}	1302.504	590.295
{random-circle, 400, 5}	1371.465	495.205
{random-circle, 400, 10}	6476.815	2299.34
{random-circle, 600, 5}	3026.25	1170.275
{random-circle, 600, 10}	11448.555	5089.92

be very close to 1 and this indeed is exactly 1 for eight of the twelve cases. Furthermore, η for PRoPHET are always significantly larger than the corresponding rates for Epidemic. Finally, the efficiency of PRoPHET is underscored by the fact that for the parameter sets in which PRoPHET exceeds the ζ values of Epidemic (cp. table IV) the η values are also significantly higher.

C. Overhead

The real overhead σ, i.e. average number of replicas per generated message or average number of hops per message, is given in Table VI. This metric seems to be linearly correlated with the particle count, as well as the scan radius. Strikingly, Epidemic produces more than 100% more overhead than PRoPHET in each instance, in several instances even 200% more. In addition the plots in both figures also illustrate the effect of an increase in scan radius resulting in an increase of the overhead metric by factors from about four to five.

Although the overhead for the random-circle mobility model is consistently higher for corresponding configurations of the random-walk mobility model, the differences are very small. This is also seen in Figures 6 and 7 as the plots for both mobility models the same scan radius are very close.

VI. Conclusion

This paper demonstrates the usefulness of the Swam Network Simulator as an OppNet simulator. The experiments are setup using the internal components that provided the necessary basics to evaluate the two routing protocols.

In general, the delivery success rates for PRoPHET are found to be far inferior to those of Epidemic. In some cases they are not even comparable. However, PRoPHET beat Epidemic in terms of delivery efficiency and overhead. Therefore, it has to be emphasized that the higher delivery success rates of Epidemic are to the disadvantage of both delivery efficiency and overhead. This partly confirms previous research of the two protocols. However, the low delivery rates of PRoPHET were not as expected and require more research, e.g. by fine tuning the internal parameters of PRoPHET. Furthermore,

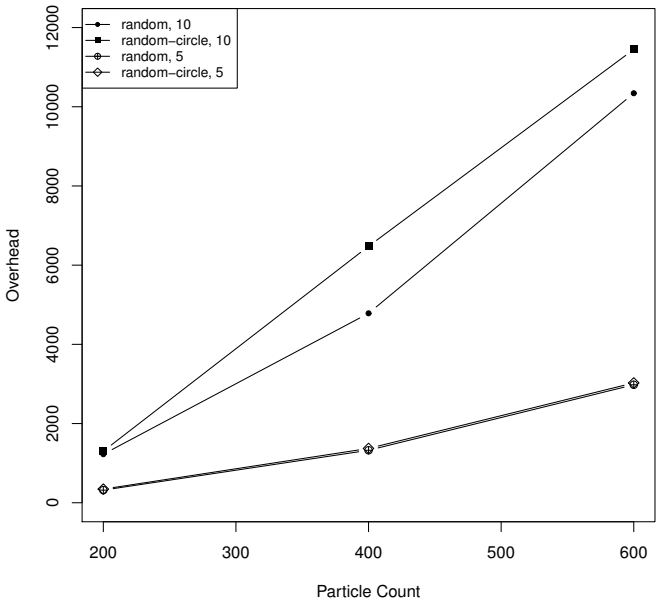


Fig. 6: Overhead σ of Epidemic as a function of particle count for the different mobility models and scan radii.

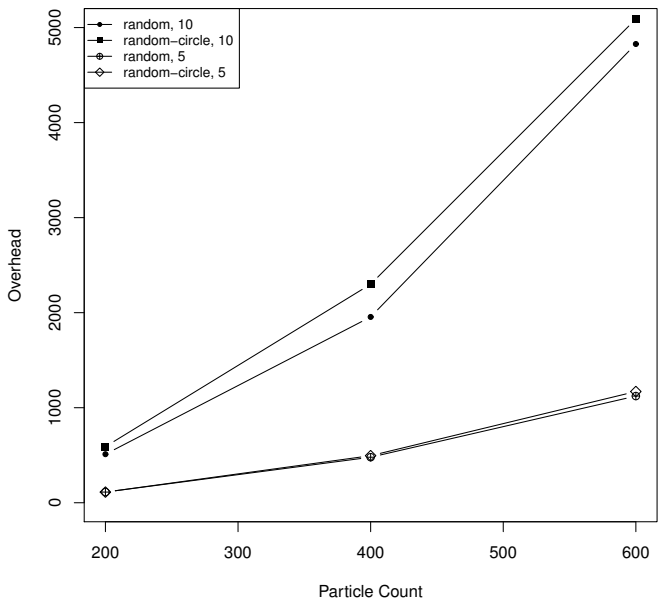


Fig. 7: Overhead σ of PRoPHET as a function of particle count for the different mobility models and scan radii.

the observation that higher scan radii negatively impact the delivery probabilities of Epidemic, require closer inspection as this seems counter intuitive. Tying this in with the previously mentioned catastrophe scenario of establishing communication in an OppNet of mobile phones, the memory requirements Epidemic routing may be a limitation. The algorithms flooding nature would enable higher delivery success as seen in the results of this paper. But the sacrifice would of course be in terms of storage - for keeping all those messages - and also battery (especially for communication between phones). Both are limited resources even on today's modern smartphones. This also somewhat applies for PRoPHET, since computing and storing estimated delivery probabilities for an encounter are costs, which are additional costs over Epidemic routing. The optimization of memory requirements may therefore be an interesting aspect for further research in the future.

In conclusion, this paper illustrates Swarm-Sim as a capable and applicable simulator for message passing in OppNets without requiring much configuration.

In future, we aim to investigate how to implement this functionality e.g. through simple devices running an Android OS. For that we investigated in [9] how to use Android devices for local, decentralized device-to-device communication. Using our findings in [10] allows to prioritize the communication in order to increase the message delivery ratio independent of the routing algorithm. Finally, ideas from [11] can be included to address security issues in mesh networks.

ACKNOWLEDGMENTS

We would like to thank the Arab-German Young Academy of Sciences and Humanities for funding this research as well as our collaboration partner Dr. Ahmed Khalil for the valuable interdisciplinary discussions on future nano robot swarms.

REFERENCES

[1] A. Cheraghi, T. Amft, S. Sati, P. Hagemeister, and K. Graffi, "The state of simulation tools for p2p networks on mobile ad-hoc and opportunistic networks," in *Proceedings of the International Conference on Computer Communications and Networks (ICCCN)*. IEEE, 2016.

[2] S. Jain, K. Fall, and R. Patra, *Routing in a delay tolerant network*. ACM, 2004, vol. 34, no. 4.

[3] A. Vahdat, D. Becker *et al.*, "Epidemic routing for partially connected ad hoc networks," Department of Computer Science, Duke University, Tech. Rep., 2000.

[4] A. Lindgren, A. Doria, and O. Schelen, "Probabilistic routing in intermittently connected networks," in *Service assurance with partial and intermittent resources*. Springer, 2004.

[5] A. Keränen, J. Ott, and T. Kärkkäinen, "The one simulator for dtn protocol evaluation," in *Proceedings of the International Conference on Simulation Tools and Techniques (SIMUtools)*. ICST, 2009.

[6] D. Vardalis and V. Tsaoussidis, "Exploiting the potential of dtn for energy-efficient internetworking," *Journal of Systems and Software*, vol. 90, 2014.

[7] A. R. Cheraghi. (2020) Swarm simulator. [Online]. Available: https://gitlab.cs.uni-duesseldorf.de/cheraghi/swarm-sim

[8] J. J. Ahmad Reza Cheraghi. (2020) Opportunistic networks for swarms. [Online]. Available: https://gitlab.cs.uni-duesseldorf.de/cheraghi/swarm-sim/-/tree/oppswarm

[9] A. Ippisch and K. Graffi, "Infrastructure Mode Based Opportunistic Networks on Android Devices," in *Proceedings of the IEEE 31st International Conference on Advanced Information Networking and Applications (AINA)*, 2017, pp. 454–461.

[10] S. Sati, C. Probst, and K. Graffi, "Analysis of Buffer Management Policies for Opportunistic Networks," in *Proceedings of the 25th International Conference on Computer Communication and Networks (ICCCN)*, 2016, pp. 1–8.

[11] P. S. Mogre, K. Graffi, M. Hollick, and R. Steinmetz, "A security framework for wireless mesh networks," *Wireless Communications and Mobile Computing*, vol. 11, no. 3, pp. 371–391, 2011.