FinalComm: Leveraging Dynamic Communities to Improve Forwarding in DTNs

Naercio Magaia, Pedro Dinis Gomes, Paulo Rogério Pereira
INESC-ID, Instituto Superior Técnico, Universidade de Lisboa
Lisbon, Portugal
naercio.magaia@tecnico.ulisboa.pt,pedro.dinis.gomes@tecnico.ulisboa.pt,prbp@inesc.pt

ABSTRACT

This article proposes a social-based routing protocol for Delay-Tolerant Networks (DTNs) known as FinalComm. FinalComm allows nodes to build a view of the existing communities in the network. With this information, nodes are able to relay messages after computing if the neighbor will most probably meet the destination or any other node from the community of the destination of the message.

Simulation results shows that FinalComm is able to achieve a high delivery rate and an extremely low overhead ratio, if compared with four other routing protocols. Specifically, FinalComm presents average gains of 31.4% in terms of delivery rate in comparison with other routing protocols for the scenarios considered.

CCS CONCEPTS

• **Networks** → *Network simulations*; *Ad hoc networks*;

KEYWORDS

Delay-Tolerant Networks, Routing protocol, Community, Algorithms

1 INTRODUCTION

Delay-Tolerant Networks (DTNs) [6] are characterized by not having permanent end-to-end connectivity and therefore presenting long and variable delays, high error rates, and intermittent connectivity. To send a message from one point to another, it can take minutes, weeks or even months. At each time frame, the connections or opportunities that one node has to transmit a message may be different due to the mobility of the nodes. Since a DTN is a dynamic network, it may be necessary for some intermediate nodes to store messages, carry them and, if adequate, forward them to a next hop node or the destination, if that is the case. This is known as the *store-carry-and-forward* approach. Bearing this in mind, it is understandable why classical routing protocols for Mobile Ad-Hoc Networks (MANETs) are not suitable for DTNs.

Some member of the DTN research community are focused on developing space communications [6] meanwhile others only work with scenarios where there are few communication's infrastructures and end-to-end connectivity does not exist all the time.

Consider, for instance, least developed countries where deploying a DTN can bring huge advantages. It is possible that people that live in poorer or remote areas where there are no communication's infrastructures or the few ones that exist are not affordable to them may benefit from it. In such cases, by having a simple device like a smart-phone, in which they just have to invest once and the cost is smaller than an annual subscription for Internet access, they can have access to emails, for example, if a DTN is deployed. It is obvious that they do not receive emails instantly, but at least they are able to have access to information more inexpensively.

Although in developed countries there are multiple efficient communication's infrastructures, DTNs can also have a useful role. Remote areas exist even in developed countries, where typically the population does not have many resources. Alternatively, imagine, for example, a natural catastrophe that brings communications down. It would be desirable to have a backup system for some services, which could continue to operate in a DTN. In summary, DTNs can be used in multiple scenarios through a fast and cheap deployment when comparing with more complex communication's infrastructures.

Hence, DTNs have many use cases. It is however important that they are deployed in an efficient manner, that is, the routing protocols proposed for DTNs should have algorithms that are able to increase the number of messages that reach their destinations using an intermittent end-to-end connection and to consume the least possible number of resources such as energy and memory from devices.

Some routing protocols have been proposed in literature [4, 11] that use social metrics such as *community*. According to sociology [9], a community can be defined as a group of people living in the same location and sharing similar interests. These routing protocols that use social metrics, also known as social-based routing protocols, present higher delivery probability if compared to those that do not use such metrics. However, the algorithms used by these protocols to detect communities only consider that nodes can join a community meanwhile they should also be able to leave them, similarly to what people eventually do when they move from one location or region into another.

This article proposes FinalComm, a new social-based routing protocol for Delay-Tolerant Networks (DTNs). FinalComm takes into consideration social relationship between nodes when it has to decide if a message should be sent from one node to another or even deleted. It also consider that social interests may change over time and proposes algorithms that capture and update the utilized social structures accordingly.

PE-WASUN'17, November 21–25, 2017, Miami, FL, USA

ACM ISBN 978-1-4503-5166-9/17/11…$15.00
https://doi.org/10.1145/3134829.3134836

The article is organized as follows. Section 2 presents background and related work. Section 3 presents the concept behind the FinalComm protocol, its architecture and mechanisms. In Section 4 and 5, the simulation model and results are presented. Finally, conclusions drawn from this work are presented in Section 6, along with a few remarks on future work.

2 BACKGROUND AND RELATED WORK

DTN routing protocols can be divided in two categories: traditional and social-based. The first category includes all routing protocols that do not consider relationships among nodes (e.g., [7, 14, 15]) meanwhile the second includes protocols that try to explore social aspects, i.e., the fact that nodes tend to have mobility patterns that are based on social factors instead of simply random.

According to [17], it is possible to enumerate at least three important social metrics that DTN routing protocols are based on: community, centrality and similarity.

One of the most important social metric is *community*. When analyzing social relationships between nodes, adding them to one or more communities is a good way of summarizing their common interests. Some other aspect to consider within a community is the level of interaction among its members. Usually, some members tend to interact more than others. It has also been proven that a member of a community will most likely interact with other members of the same community than with strangers [16].

Another important social metric is *centrality*, which represents a measure of how important a node is in a network. This importance can be categorized in three different groups: degree centrality, betweenness centrality and closeness centrality [10]. The first, and the simplest, refers to the number of links a node has and it can be calculated locally. Usually, nodes with the highest number of links are the ones with which other nodes want to forward messages to. Due to their high number of connections, it is most likely that the messages will reach their destination faster. The second, refers to the number of shortest paths that crosses a given node overall shortest paths. In the last type, the centrality of a node is calculated by inversing the average shortest path distance from it to all other nodes.

Similarity describes how close two nodes are from each other. It allows estimating the probability of two node to re-encounter in the future considering: the number of common neighbors between them, the number of shared interests or the number of shared locations.

Now, some relevant social-based routing protocols to this work are presented. The BubbleRap [4] routing protocol is based on two social metrics, community and centrality, to reduce the number of copies of the same message around the network but having a high delivery probability. It is based on Label [3] and Rank [1] algorithms. The Label algorithm uses explicit labels in nodes to identify the communities they belong to, while the Rank algorithm forwards messages to nodes with higher centrality than the current node. Generally speaking, a node belongs to a community and it is assumed that each node is part of at least one community. On the other hand, to identify the most popular nodes, the protocol uses betweenness centrality that is difficult to estimate in dynamic networks [10].

As stated in [4], both Label and Rank algorithms have some limitations. The Label algorithm is not capable of forwarding messages away from the source when the destination is socially far away, and the Rank algorithm, in which each node does not have a view of the global ranking, is not appropriate for big scenarios, as small communities will be difficult to reach. The BUBBLE algorithm was proposed to overcome the previous drawbacks. BUBBLE works as follows: if a node wants to send a message to a certain destination, the first thing that it does is to bubble this message to another node that has a higher global rank until the message reaches a node with the same label of the destination node's community. After that, global ranking will be no longer used and instead messages are bubbled up using local ranks. In the end, messages will successfully reach their destinations or they will expire. Communities are detected by one of two possible ways: labels or a distributed mechanism. The k-clique algorithm and Weighted Network Analysis (WNA) [12] are viable options as a distributed mechanism. Based on [13], a k-clique community is defined by a union of complete sub-graphs of size k (k-cliques). Two k-cliques are adjacent if they share k-1 nodes. The k-clique algorithm was designed for binary graphs but with a threshold for edges of the contact graphs, therefore it can be used in DTN routing protocols [4].

The core of the BubbleRap protocol is an algorithm called DiBuBB, which is a modified version of BUBBLE. In it, BUBBLE is implemented in a distributed way and the mechanism used for detecting communities is the k-clique algorithm, which has detection accuracy up to 85%.

The dLife [11] routing protocol was proposed to explore a limitation of several other approaches that did not consider that social structures could evolve over time. It explores the user's daily life routine, as they can be used to predict future interaction between nodes. The social metric used is similarity. The dLifeComm [11] routing protocol, which is slightly different from dLife as it takes into account communities, was also proposed.

One of the drawbacks of the previous social-based protocols is when the destination is part of a community in which every node has a low global rank. In this case, proper relay nodes may be difficult to identify. In addition, these protocols only consider that nodes can join a community meanwhile they should also be able to leave them.

3 THE FINALCOMM PROTOCOL

FinalComm is a social-based routing protocol for DTNs that uses two social metrics, namely community and similarity. It addresses a drawback related to the use of global centrality of the nodes meanwhile also considering that social interests may change over time, i.e., nodes can also be removed from communities.

The main idea behind this approach is that nodes should have a bigger view of the network, even though that view could not always be the most updated one. Therefore, nodes have to store other node's communities, which is done every time they meet. Let node A and node B be in communication range between each other. Node A will store node B's community since it is the most updated information that node A can get from node B, and vice-versa. Then, both nodes will update their communities' information considering the most recent information received. In FinalComm,

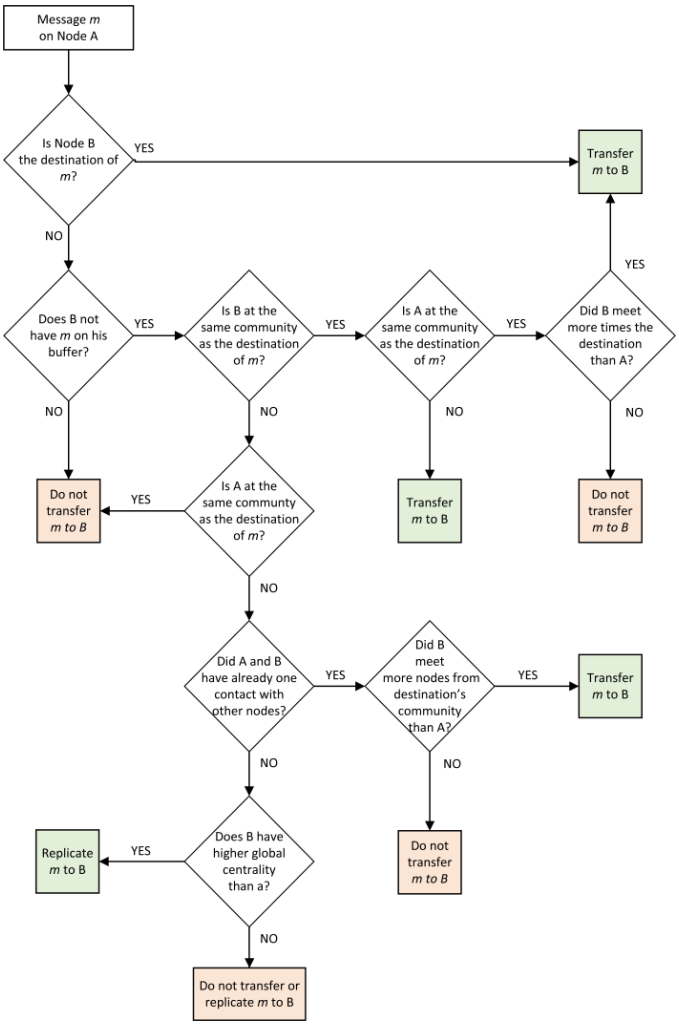


Figure 1: The block diagram of FinalComm

node B belongs to node A's community if the total contact time between A and B is greater or equal to the community familiar threshold (hereafter the *commfamiliar* threshold), considering all the contact time with all other nodes in the network that A has had. Hence, it is expected an increase in the forwarding efficiency by taking into account the community of the destination of the message. It is also expected that the number of successful delivered messages (Eq. 1) and overhead ratio (Eq. 2) to increase and decrease, respectively.

$$delivery\ rate = \frac{messages\ delivered}{messages\ created} \tag{1}$$

$$overhead\ ratio = \frac{messages\ relayed - messages\ delivered}{messages\ delivered} \tag{2}$$

The core algorithms of FinalComm determine (*i*) when to send a message to another node and (*ii*) when a node should delete a message that was sent. Other two important algorithms are part of the process of (*iii*) creating and (*iv*) updating communities. All of them should be seen from a single node's perspective. Please note that all these algorithms assume that nodes are fully cooperative.

3.1 Algorithms to send or delete messages

Below, two algorithms used to determine when to send a message to another node and when to delete a message that was sent are presented.

`Algorithm 1: When should a node send a message to another node?` This algorithm is called when two nodes meet. It is responsible for checking if the messages that are stored in the nodes' buffers should be sent to the other node.

Consider, for instance, that node A meets node B and A has messages in its buffer. For every message m, node A uses this algorithm to take forwarding decisions. *First*, node A checks (or verifies) if node B is the destination of m. If so, it forwards m to B provided that B does not already has it. If B is not the destination, A performs other verifications. This first verification is also performed by other routing protocols such as BubbleRap, dLife and dLifeComm. *Second*, node A verifies whether B has already m in its buffer. If B has m on its buffer, A does not send m to B. Otherwise, A continues the verification. This second verification is also performed by dLife and dLifeComm. *Third*, A verifies whether just one of the messages belongs to the same community as the destination node of m. If it is B, A sends m to B, however if it is A, A keeps m in its buffer. This third verification is also performed BubbleRap. *Forth*, if both A and B belong to the same community as the destination node of m, the number of times that each one has met with the destination is computed. Node A only forwards m to B if B has encountered the destination more times than it. If both A and B belong to different communities and neither of these communities is the same as the community of the destination node of m, it is computed the number of times that each node has met nodes from the destination's community. Node A only forwards m to B, if A has met that community less times. The forth step is new and therefore different from the other protocols, and is considered the core of the FinalComm protocol. *Last*, and similarly to BubbleRap, the global centrality (degree centrality) of the nodes is computed. After that computation, node A replicates m to B if B has higher global centrality than it. Otherwise, A decides just to keep m in its buffer. Please note that the latter is only used if both nodes belong to different communities that are not the same as the destination node's community, and at least one of the nodes did not have any contacts with other nodes. That is, it is only used in the beginning of and not many times thereafter.

Figure 1 shows a block diagram of `Algorithm 1`.

`Algorithm 2: When should a node delete a message that was sent?` After a node have decided that it should send a message to another node, it has to verify if that exact message needs to be deleted or not from its buffer. In FinalComm, a node will only delete a message from its buffer if that message has been relayed by any of the mechanisms described in `Algorithm 1` except when the message is relayed using the global centrality feature. In that case, the node will never delete the message from its buffer. It is important to mention this mechanism because it is partially

responsible for FinalComm's overhead. However, a message will also be deleted from a buffer if it reaches the Time-to-Live (TTL) established or if a new message arrives and the buffer is full. In this last case, older messages in the buffer are dropped until there is enough room for the new one.

3.2 Algorithms to create and update of communities

As previously mentioned, previous work [4, 11] leveraged on the k-clique algorithm to create communities. However, they do not consider that communities can change over time, that is, they only consider that nodes can join a community meanwhile they should also be able to leave them. In order to better describe what happens in real life, a different although simpler, mechanism to build communities and keep them updated was proposed and implemented in FinalComm.

The mechanism works as follows: when two nodes meet, before they start exchanging messages they need to do some verifications and to share information about the communities they know. The mechanism is composed of two algorithms that are presented below:

`Algorithm 3: The Creation of Communities`. In this algorithm, nodes A and B start by verifying if each node that belongs to their communities should remain in the community by means of the *commfamiliar* threshold and by taking into account the percentage of contact time between each one of them and the node. If it happens that the node does not meet this threshold criterion, nodes A and B remove the node from their communities. Then, each node does a similar check to decide if the other node that it met should be part of its community. To do so, nodes A and B consider the percentage of contact time between them and the node, and compare the result with the $comm\,familiar$ threshold. Node A only adds B if that result is greater or equal to the threshold and vice-versa. Finally, nodes change their network's view and the most recent data will be kept by each one in their known communities. To see who has the most recent view of a community in their known communities, nodes A and B verify the last time that that information was updated. Older data is deleted. After some time, a connection between two nodes will be lost. When that happens, it is necessary to update some information about each node's perspective of its community, which is done by the next algorithm.

`Algorithm 4: The Updating of Communities`. The process of updating communities is identical to what happens on the previous algorithm when calculating the percentage of contact time between nodes A and B and another node. In `Algorithm 4`, node A verifies whether B should be added or remain in its community. It can happen that node A sees node B as part of its community, however node B does not due to having different contacts with other nodes. Therefore, their total contact time with nodes will be different. However, this can describe what happens in some online social networks nowadays. For example, on Facebook one person can be followed by another without adding or also following her back, and still be able to interact with each other.

3.3 The dynamic parameter tuning approach

Social-based routing protocols that use the community metric have some parameters related to the way cluster or communities are built that need to be tuned in order to achieve the best possible performance. However, it is only possible to detect the best values for these parameters if multiple simulations are performed, which is not practical in most cases. Moreover, even if these parameters are found, all nodes will be using those same values without exceptions. What if these values are not the most adequate ones for every node?

For FinalComm, the parameter that requires tunning is the *commfamiliar* threshold. In order to understand if there is the possibility to create a dynamic mechanism to adjust the *commfamiliar* parameter, minor changes were made to `Algorithm 3` that is responsible for creating communities. In this new approach, nodes will start by having the *commfamiliar* threshold set to a certain value that works well. Then, the goal is to give nodes a chance to regularly change this threshold after a period of time. Two different periods of update time were considered: half a day and a day. In practice, when the update time comes, a node will check: how many nodes it actually has on its community and how many other nodes it knows that exist in the network. Then, with that information, the node computes its *community view* metric that is given by

$$community\ view = \frac{nodes\ in\ community}{total\ nodes\ knwon} \times 100$$

If this metric is inferior to a certain percentage, the *commfamiliar* threshold should be increased in order to add more nodes to its community. If the metric is superior to a certain percentage, the number of nodes in its community should decrease and therefore the *commfamiliar* threshold will be increased. The ranges of percentages considered to decide if there is any change are 15%-20%, 20%-25%, 25%-30%, 30%-35% and 35%-40%. When the *commfamiliar* threshold is being updated, if its value is inferior to 2.0%, the increments and the decrements will be 0.1%. Otherwise, 1.0% will be used. Therefore, the *commfamiliar* threshold can never take values less than 0.1% and greater than 100%.

4 SIMULATION MODEL

FinalComm was implemented in the Opportunistic Network Environment (ONE) simulator [5]. Different simulation scenarios consisting of two synthetic mobility models were considered. It is assumed here, as in most networks of interest, that there is some social structure between the nodes participating in the network. The simulation time was set to 11.5 days with an update interval of 1.0 s. The warmup time was set to 1 day. The warmup time corresponds to a simulation interval, from the beginning of the simulation, in which the network's statistics are not taken into account. This particular warmup time of one day was chosen to analyze if protocols that use the community property and the ones that are based on daily routines could benefit from it. It is expected that one day will be enough to ensure that communities are already created and daily routines are detected.

The message size was set to 200 kB. Only two nodes within range could communicate with each other at a time. The communication range between nodes was 10 m, and the communication was bidirectional at a constant transmission rate, for Bluetooth

and Wi-Fi interfaces, of 2 Mbit/s and 10 Mbit/s, respectively. Every 60 s, a source node randomly chosen generated one message to a randomly chosen destination.

The following mobility models were considered:

Shortest-path Map-Based Movement (SPMBM). SPMBM consisted of a network with 80 pedestrians, 40 cars and 6 trams over the downtown area of Helsinki, Finland. Pedestrians were moving at a speed varying between 0.5 to 1.5 m/s. Cars and trams were moving at a speed varying between 2.7 to 13.9 m/s and 7 to 10 m/s, respectively. The TTL attribute of each message was 5 h. The pedestrians and cars had a buffer size of 5 MB. Trams had a buffer size of 50 MB for DTN traffic.

Working Day Movement (WDM). WDM [2] consisted of a network with 150 pedestrians and 20 buses over the midtown area of Manhattan, United States of America. There were 10 offices and the working day length was 8 h. The probability of going shopping after work was 50% and there were 20 meeting points. Pedestrians and buses were moving at a speed varying between 0.8 to 1.4 m/s and 7 to 10 m/s, respectively. The TTL attribute of each message was approx. 24 h. All nodes had a buffer size of 5 MB for DTN traffic.

5 SIMULATION RESULTS

In this section, several simulation results describing the performance of FinalComm are presented. For each setting, i.e., protocol-configuration parameter pair, five independent simulations using different message generation seeds were conducted, and the results averaged, for statistical confidence. FinalComm was compared with well-known DTN routing protocols [8]: one traditional routing protocol, namely Epidemic, and three social-based routing protocols, namely BubbleRap, dLifeComm and dLife.

The performance of FinalComm was evaluated according to the following metrics: delivery ratio, overhead ratio, average latency and average hop count. The delivery ratio is a key performance indicator as it tells the percentage of successfully received packets of all sent. The overhead ratio is the number of message transmissions for each delivered message. The average latency corresponds to the average time that a message takes from the node that creates it to its destination node. The average hop count corresponds to the average number of hops that a message needs to take before reaching the destination.

The dynamic parameter tuning approach (hereafter dynamic mechanism) will be also evaluated aiming at understanding if nodes that are using FinalComm benefit from having different values of the *commfamiliar* threshold and from updating this threshold automatically.

In protocols that uses the community metric to relay messages, it will be also analyzed the number of communities and the average number of nodes per community at the end of each simulation.

5.1 SPMBM

Figure 2 compares the performance of all the routing protocols considered in terms of delivery rate, overhead ratio, average latency and average hop count for the SPMBM scenario. For each protocol, the best performance values were selected for the comparison. There is also a comparison between the performance obtained with (i.e., Normal) and without warmup time (i.e., Warmup).

In the SPMBM scenario, FinalComm was able to deliver more messages than the others routing protocols. In fact, the difference is significant. FinalComm was able to deliver 13% more messages than dLife, which came second in terms of delivery rate (see Figure 2(a)). The reason why dLife and dLifeComm did not perform better than FinalComm might have been related to the fact that there were no daily routines in this scenario. One could argue that this scenario is not the ideal one for those two protocols.

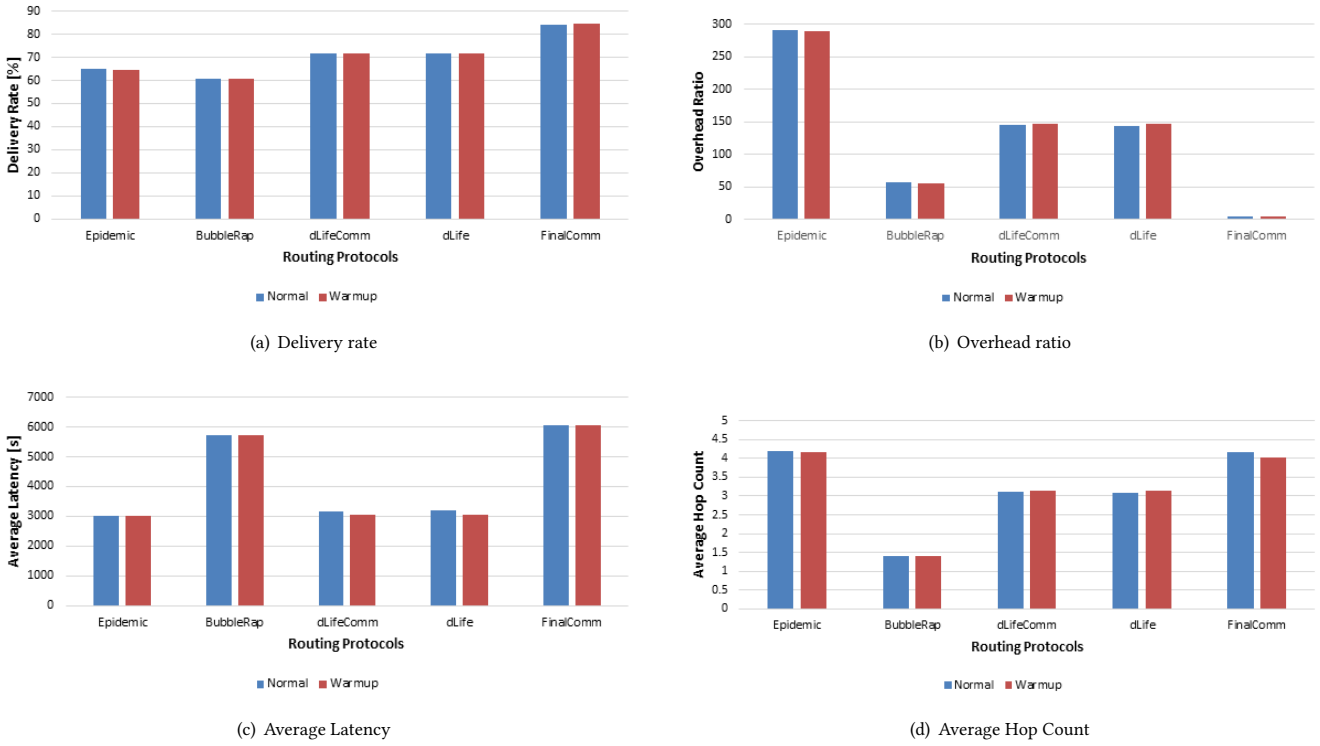
In terms of overhead ratio (see Figure 2(b)), FinalComm also performed considerable better than all the other protocols. It only needed on average approximately 5 replicas of each delivered message to achieve the highest delivery rate meanwhile BubbleRap, which comes second, needed almost 60 message replicas. There is also a huge difference between BubbleRap and dLifeComm in terms of overhead ratio. Both protocols use the same algorithm to build communities and this algorithm can be in part responsible for the overhead. It is important to remember that the k-clique algorithm does not remove nodes from communities. Aside from the way protocols decide how to forward messages, how they decide to delete them can also significantly influence the protocol's overhead. Overall, FinalComm tends to be very precise in the way it deletes messages. The way all the proposed algorithms work together within FinalComm allowed it achieve the highest delivery rate with the smallest overhead ratio.

From Figure 2(b) and Figure 2(c) it is possible to see an interesting fact that also supports the findings on the last paragraph. With the exception of Epidemic, the order in which protocols are placed in terms of how high the average latency are, corresponds to the inverse order on how low the overhead ratios are, even though this relation is not proportional. For achieving a low overhead, the messages may have to spend more time at nodes' buffer, i.e., routing protocols should have mechanisms to ensure that the next relay node that receives the message will be better than the current one and in that case the messages could be deleted more often from the buffers.

When comparing Figure 2(a) to Figure 2(d), the order in which protocols appear is the same, except for Epidemic. For a high delivery rate comes a higher number of hops. At first sight, more hop counts could lead to more overhead, but as state before, the algorithms used by FinalComm to relay and delete messages ensures that that does not happen. Therefore, more hops can lead to a better delivery rate, hence to have more hops the overhead should be controlled. Otherwise, more overhead will lead to less space in buffers and more removed messages, which affects the delivery rate.

In none of the presented figures there was a case in which the results for a warmup time of one day, which is significant, would benefit the protocol. One may conclude that communities are formed very fast and protocols can efficiently relay messages even with partially formed communities.

Table 1 presents the number of detected communities and the average number of nodes per community at the end of simulations of three different protocols, namely BubbleRap, dLifeComm and FinalComm. In BubbleRap there are many communities, although each has almost on average 1 node meanwhile in dLifeComm the



(a) Delivery rate

(b) Overhead ratio

(c) Average Latency

(d) Average Hop Count

Figure 2: Delivery rate, overhead ratio, average latency and average hop count for all the routing protocols considered in the SPMBM scenario

Table 1: The number of communities and the average number of nodes per community for the SPMBM scenario

Protocol	Number of Communities	Average Nodes per Community
BubbleRap	115.00 ± 2.77	1.22 ± 0.56
dLifeComm	1.00 ± 0.00	126.00 ± 0.00
FinalComm	126.00 ± 0.65	105.20 ± 18.37

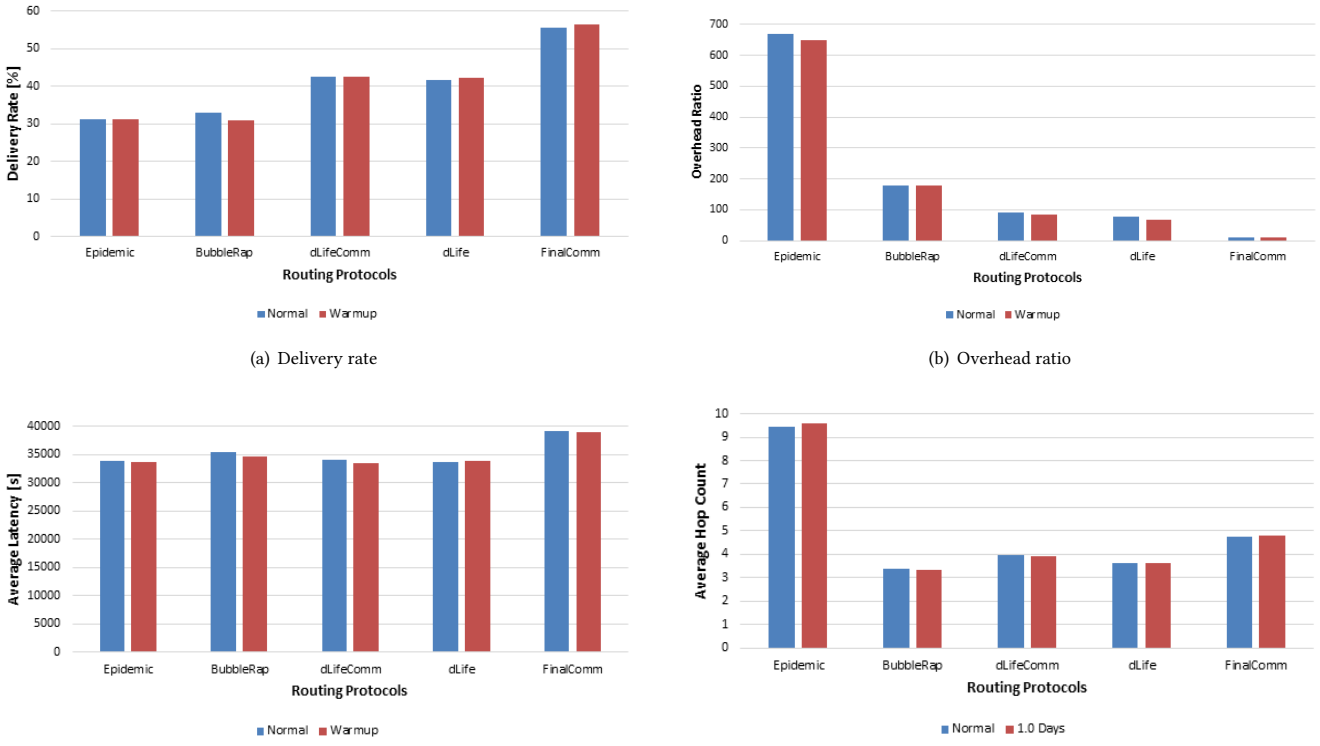
opposite happens. The latter case explains why the k-clique algorithm should have a feature to remove nodes from communities. Depending on the parameters such as k, it may only exist one community with all the nodes or many communities composed by a single node each. Since FinalComm uses a different algorithm to build communities, every node has a unique community with some nodes.

Finally, in order to evaluate the dynamic mechanism, the *commfamiliar* threshold was initially set to 0.50%. The best case was detected when the update time took the value of 43200 seconds and when the nodes were trying to populate their communities with 35 to 40% of all existing nodes. In this case, the dynamic mechanism was able to achieve a delivery rate of 84.15% with an overhead of 10.37. The dynamic mechanism was able to deliver an identical number of messages compared to the best case of FinalComm without this mechanism. However, the overhead increased approximately 100%. The latter was related with the constant changing in the *commfamiliar* threshold values, which introduced one more hop in the average hop count. Overall, the mechanism performed well in this scenario.

5.2 WDM

Figure 3 compares the performance of all the routing protocols considered in terms of delivery rate, overhead ratio, average latency and average hop count for the WDM scenario. For each protocol, the best performance values were selected for the comparison. There is also a comparison between the performance obtained with (i.e., Normal) and without a warmup time (i.e., Warmup).

Even in scenario where the movement patterns represented daily routines, FinalComm was able to deliver more messages than all the others routing protocols considered (see Figure 3(a)). FinalComm delivered approximately 23% more messages than dLifeComm, which came second in terms of delivery rate. With or without daily routines, the proposed protocol outperformed all the other routing protocols.



(a) Delivery rate

(b) Overhead ratio

(c) Average Latency

(d) Average Hop Count

Figure 3: Delivery ratio, overhead ratio, average latency and average hop count for all the routing protocols considered in the WDM scenario

Table 2: The number of communities and the average number of nodes per community for the WDM scenario

Protocol	Number of Communities	Average Nodes per Community
BubbleRap	170.00 ± 0.00	51.89 ± 50.29
dLifeComm	41.50 ± 20.63	162.10 ± 21.02
FinalComm	170.00 ± 0.00	34.98 ± 18.55

In terms of overhead, FinalComm also performed much better than the other protocols. It only needed on average approximately 9 replicas of each delivered message to achieve the delivery rate mentioned above meanwhile dLife, which comes second, needed almost 79 replicas. All the previous results refer to cases in which no warmup time was considered. dLifeComm came in third place with an overhead really close to that presented by dLife. Then, there is also a big difference between BubbleRap and dLifeComm in terms of overhead, i.e., twice in terms of magnitude. As expected, in the last place with a huge overhead was Epidemic. Its overhead ratio was approx. 667.

From Figure 3(a) and Figure 3(b), it is possible to see that for the highest delivery rate comes a low overhead. Even though the results are not proportional, the order in which protocols come in terms of better delivery rate is approximately the opposite of the overhead ratio.

From Figure 3(a), Figure 3(c) and Figure 3(d), one may conclude that the highest delivery rate achieved had the highest average latency and the highest average hop count associated, excluding the case of Epidemic. Figures 3(c) and 3(d) show that messages in FinalComm took more time at nodes' buffers and also crossed more nodes on average. Thus, by having a good mechanism to control the protocol's overhead, crossing more nodes does not necessarily brings the delivery rate down. Epidemic is the best example to show how important is to control the overhead.

In general, no protocol performed well in the WDM scenario since the delivery rate was too low if compared to the SPMBM scenario. Approximately, only half of the messages were delivered in the best cases identified. A way to improve all the delivery rates could be to increase both messages' TTL and nodes' buffers.

Similarly to the SPMBM scenario, in none of the presented figures there was a case in which the results for a warmup time of one day, which is significant, benefited the protocol.

Table 2 presents the number of detected communities and the average number of nodes per community at the end of the simulations of three different protocols, namely BubbleRap, dLifeComm and FinalComm.

In BubbleRap and FinalComm, every node had its unique community and the average number of nodes per community was 51.89 and 34.98, respectively. However, the confidence interval related to the average number of nodes per community in BubbleRap seemed to indicate that there were some communities with a very low and high number of nodes. In dLifeComm, there was only an average number of 41.50 communities and each one had almost all the existing nodes in this scenario.

Similarly to SPMBM, to evalute the dynamic mechanism, the *commfamiliar* threshold was initially set to 0.50%. The best case was also detected when the update time took the value of 43200 seconds, and when the nodes were trying to populate their communities with 35 to 40% of all existing nodes. In this case, the dynamic mechanism was able to achieve a delivery rate of 55.54% with an overhead of 11.33. When comparing FinalComm with and without this mechanism, it is possible to see that the delivery rates and overhead ratio were similar.

6 CONCLUSIONS

This article proposes FinalComm, which is a social-based routing protocol for DTNs. In general, FinalComm outperformed other routing protocols for the scenarios and routing metrics considered, that is, it was always the protocol with the highest delivery rates and lowest overhead ratio, even in scenario consisting of daily routines. FinalComm presented average gains of 20.86% and 41.88% in terms of delivery ratio in comparison with other routing protocols for the SPMBM and WDM scenarios, respectively.

In FinalComm, a node delivered more messages if the communities were not small. Simulation results showed that nodes should have a community composed by at least 30% and 7% of existing nodes in order to achieve the best performance possible for the SPMBM and WDM scenarios, respectively. However, a drawback of reducing the number of nodes per community was that each node had to carry messages for a longer time.

Simulation results also showed that the the dynamic mechanism was able to achieve similar results when compared to the best cases without the mechanism. However, the overhead increased significantly, although being that the price to pay. The dynamic mechanism performed better when the update time was adjusted every half a day with percentage interval of 35% to 40%, which corresponds to the fraction of total nodes a community should contain. It is also important to say that the node's *commfamiliar* threshold only converged to its final value after some days. Overall, the dynamic mechanism was a good approach and it showed that nodes do not have to share the same value of the *commfamiliar* threshold in order to achieve a good protocol's performance.

An extensive analysis of the dynamic mechanism with multiple other time periods and percentage intervals in several other scenarios was left for future work. It would be also interesting to analyse the influence of the load and the number of nodes on the scenarios considered. An evaluation of the performance of FinalComm in other scenarios such as those based on real traces was also left for future work.

ACKNOWLEDGMENTS

This work was partially supported by Fundação Calouste Gulbenkian and by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2013.

REFERENCES

[1] Lada A. Adamic, Rajan M. Lukose, Amit R. Puniyani, and Bernardo A. Huberman. 2001. Search in power-law networks. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics* 64, 4 II (sep 2001), 461351–461358. DOI: https://doi.org/10.1103/PhysRevE.64.046135 arXiv:cs/0103016

[2] Frans Ekman, Ari Keränen, Jouni Karvo, and Jörg Ott. 2008. Working Day Movement Model. In *Proceedings of the 1st ACM SIGMOBILE Workshop on Mobility Models (MobilityModels '08)*. ACM, New York, NY, USA, 33–40. DOI: https://doi.org/10.1145/1374688.1374695

[3] Pan Hui and Jon Crowcroft. 2007. How small labels create big improvements. In *Proceedings - Fifth Annual IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops 2007*. 65–70. DOI: https://doi.org/10.1109/PERCOMW.2007.55

[4] Pan Hui, Jon Crowcroft, and Eiko Yoneki. 2011. BUBBLE Rap: Social-Based Forwarding in Delay-Tolerant Networks. *IEEE Transactions on Mobile Computing* 10, 11 (Nov. 2011), 1576–1589. DOI: https://doi.org/10.1109/TMC.2010.246

[5] Ari Keränen, Jörg Ott, and Teemu Kärkkäinen. 2009. The ONE simulator for DTN protocol evaluation. In *Proceedings of the 2nd international conference on simulation tools and techniques*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 55.

[6] Maurice J Khabbaz, Chadi M Assi, and Wissam F Fawaz. 2012. Disruption-Tolerant Networking: A Comprehensive Survey on Recent Developments and Persisting Challenges. *IEEE Communications Surveys & Tutorials* 14, 2 (jan 2012), 607–640. DOI: https://doi.org/10.1109/SURV.2011.041911.00093

[7] Anders Lindgren, Avri Doria, and Olov Schelén. 2007. *Probabilistic routing in intermittently connected networks*. Technical Report. RFC 6639. http://datatracker.ietf.org/doc/rfc6639

[8] Naércio Magaia, Carlos Borrego, Paulo Pereira, and Miguel Correia. 2017. PRIVO: A PRIvacy-preserVing Opportunistic routing protocol for Delay-Tolerant Networks. In *IFIP Networking 2017*. http://dl.ifip.org/db/conf/networking/networking2017/1570333245.pdf

[9] Naércio Magaia, Alexandre P. Francisco, Paulo Pereira, and Miguel Correia. 2015. Betweenness centrality in Delay Tolerant Networks: A survey. *Ad Hoc Networks* 33 (2015), 284 – 305. DOI: https://doi.org/10.1016/j.adhoc.2015.05.002

[10] Naércio Magaia, Alexandre P. Francisco, Paulo Pereira, and Miguel Correia. 2015. Betweenness centrality in Delay Tolerant Networks: A survey. *Ad Hoc Networks* 33 (2015), 284 – 305. DOI: https://doi.org/10.1016/j.adhoc.2015.05.002

[11] Waldir Moreira, Paulo Mendes, and Susana Sargento. 2012. Opportunistic routing based on daily routines. In *2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, WoWMoM 2012 - Digital Proceedings*. IEEE, 1–6. DOI: https://doi.org/10.1109/WoWMoM.2012.6263749 arXiv:arXiv:1407.8368v1

[12] Mark EJ Newman. 2004. Analysis of weighted networks. *Physical Review E* 70, 5 (2004), 056131. DOI: https://doi.org/10.1103/PhysRevE.70.056131

[13] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. 2005. Uncovering the overlapping community structure of complex networks in nature and society. *Nature* 435, 7043 (jun 2005), 814–818. DOI: https://doi.org/10.1038/nature03607 arXiv:physics/0506133

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

[15] Amin Vahdat and David Becker. 2000. *Epidemic routing for partially connected ad hoc networks*. Technical Report. Technical Report CS-200006, Duke University.

[16] K. Wei, X. Liang, and K. Xu. 2014. A Survey of Social-Aware Routing Protocols in Delay Tolerant Networks: Applications, Taxonomy and Design-Related Issues. *IEEE Communications Surveys Tutorials* 16, 1 (First 2014), 556–578. DOI: https://doi.org/10.1109/SURV.2013.042313.00103

[17] Ying Zhu, Bin Xu, Xinghua Shi, and Yu Wang. 2013. A survey of social-based routing in delay tolerant networks: positive and negative social effects. *IEEE Communications Surveys & Tutorials* 15, 1 (2013), 387–401.