Routing in Intermittently Connected Networks using a Probabilistic Approach

Anders Lindgren† Avri Doria∗ Olov Schelén†
dugdale@sm.luth.se avri@acm.org olov@sm.luth.se

Abstract— In this paper, the problem of routing in intermittently connected networks is addressed. In such networks there is no guarantee that a fully connected path between source and destination exists at any time, rendering traditional routing protocols unable to deliver messages between hosts. There do, however, exist a number of scenarios where connectivity is intermittent, but where the possibility of communication still is desirable. Thus, there is a need for a way to route through networks with these properties. We propose PROPHET, a probabilistic routing protocol for intermittently connected networks and compare it to the earlier presented Epidemic Routing protocol through simulations. We show that PROPHET is able to deliver more messages than Epidemic Routing with a lower communication overhead.

I. INTRODUCTION

One of the most basic requirements for “traditional” networking, which also holds for ad hoc networking, is that there must exist a fully connected path between communication endpoints for communication to be possible. There are, however, a number of scenarios where this is not the case (thus rendering the use of ad hoc networking protocols impossible), but where it still is desirable to allow communication between nodes.

One way to enable communication in such scenarios, is by allowing messages to be buffered for a long time at intermediate nodes, and to exploit the mobility of those nodes to bring messages closer to their destination by transferring messages to other nodes as they meet. Figure 1 shows how the mobility of nodes in such scenarios can be used to eventually deliver a message to its destination. In this figure, node A has a message (indicated by the node being shaded) to be delivered to node D, but there does not exist a path between nodes A and D. As shown in subfigures a)-d), the mobility of the nodes allow the message to first be transferred to node B, then to node C, and finally node C moves within range of node D and can deliver the message to its final destination.

The kind of communication networks addressed in this paper are only viable for applications that can tolerate long delays and are able to deal with extended periods of being disconnected. Still, there exist many different scenarios and situations where communication is inherently intermittent, and in which it is of high interest to develop methods of communication despite the lack of omnipresent connectivity. Such scenarios occur in a very wide range of locations and situations.

The aboriginal Saami population of reindeer herders in the north of Sweden follow the movement of the reindeer and when in their summer camps, no fixed infrastructure is available. Still, it would be desirable to be able to communicate with the rest of the world through, for example, mobile relays attached to snowmobiles and ATVs [1]. Similar problems exist between rural villages in India and other regions on the other side of the digital divide. The DakNet project [2] has deployed store-and-forward networks connecting a number of villages through relays on buses and motorcycles in India and Cambodia.

In sensor networks, a large number of sensors are usually deployed in the area in which measurements should be done. To ensure connectivity among the sensors and to get measurements from the entire area, it is common to deploy a very large number of sensors. If sensors can be mobile and transitive communication techniques can be used between them, the amount of sensors required can be reduced, and new areas where regular sensor networks have been too expensive or difficult to deploy, can be monitored. Experiments have been done with attaching sensors to seals [3], or whales [4] to get oceanografic sensor readings, or to zebras [5] to measure things on the savannah.

Previous work [6] have tried to either solve this routing problem by epidemically spreading the information through the network. We propose the use of probabilistic routing [7], using an assumption of non-random mobility of nodes to improve the delivery rate of messages while
keeping buffer usage and communication overhead at a low level.

This paper presents a framework for probabilistic routing in intermittently connected networks. A probabilistic metric called delivery predictability is defined. Further, it defines a probabilistic routing protocol using the notion of delivery predictability, and evaluates it through simulations versus the previously proposed Epidemic Routing [6] protocol.

The rest of the paper is organized as follows. Section II describes some related work and in Sect. III, our proposed scheme is presented. In Sect. IV the simulation setup is given, and the results of the simulations can be found in Sect. V. Finally, Sect. VI discusses some issues and looks into future work and Sect. VII concludes.

II. RELATED WORK

A. Epidemic Routing

Vahdat and Becker present a routing protocol for intermittently connected networks called Epidemic Routing [6]. This protocol relies on the theory of epidemic algorithms by doing pair-wise information of messages between nodes as they get contact with each other to eventually deliver messages to their destination. Hosts buffer messages even if no path to the destination is currently available. An index of these messages, called a summary vector, is kept by the nodes, and when two nodes meet they exchange summary vectors. After this exchange, each node can determine if the other node has some message that was previously unseen to this node. In that case, the node requests the messages from the other node. This means that as long as buffer space is available, messages will spread like an epidemic of some disease through the network as nodes meet and “infect” each other.

Messages contain a hop count field, which is similar to the TTL field in IP packets and determines the maximum number of hops a message can be sent, and can be used to limit the resource utilization of the protocol. Messages with a hop count of one will only be delivered to their final destination.

The resource usage of this scheme is regulated by the hop count set in the messages, and the available buffer space at the nodes. If these are sufficiently large, the message will eventually propagate throughout the entire network if the possibility exists. Vahdat and Becker show that by choosing an appropriate maximum hop count, rather high delivery rates can be achieved, while the required amount of resources can be kept at an acceptable level in the scenarios used in their evaluation [6].

III. PROBABILISTIC ROUTING

Although the random way-point mobility model is popular to use in evaluations of mobile ad hoc protocols, real users are not likely to move around randomly, but rather move in a predictable fashion based on repeating behavioral patterns such that if a node has visited a location several times before, it is likely that it will visit that location again.

In the previously discussed mechanisms to enable communication in intermittently connected networks, such as Epidemic Routing, very general approaches have been taken to the problem at hand. There have, however, not been any attempts to make use of assumed knowledge of different properties of the nodes in the network in a truly distributed way.

We note that in an environment where buffer space and bandwidth are infinite, Epidemic Routing will give an optimal solution to the problem of routing in an intermittently connected network with regard to message delivery ratio and latency. However, in most cases neither bandwidth nor buffer space is infinite, but instead they are rather scarce resources, especially in the case of sensor networks. Therefore, it would be of great value to find an alternative to Epidemic Routing, with lower demands on buffer space and bandwidth, and with equal or better performance in cases where those resources are limited, and without loss of generality in scenarios where it is applicable.

A. PROPHET

To make use of the observations of the non-randomness of mobility and to improve routing performance we consider doing probabilistic routing and propose PROPHET, a Probabilistic ROuting Protocol using History of Encounters and Transitivity.

To accomplish this, we establish a probabilistic metric called delivery predictability, \( P_{(a,b)} \in [0,1] \), at every node \( a \) for each known destination \( b \). This indicates how likely it is that this node will be able to deliver a message to that destination. The operation of PROPHET is similar to that of Epidemic Routing. When two nodes meet, they exchange summary vectors, and also a delivery predictability vector containing the delivery predictability information for destinations known by the nodes. This additional information is used to update the internal delivery predictability vector as described below. After that, the information in the summary vector is used to decide which messages to request from the other node based on the forwarding strategy used (as discussed in Sect. III-A.2).

A.1 Delivery predictability calculation

The calculation of the delivery predictabilities have three parts. The first thing to do is to update the metric whenever a node is encountered, so that nodes that are often encountered have a high delivery predictability. This calculation is shown in Eq. 1, where \( P_{init} \in (0,1] \) is an initialization constant.

\[
P_{(a,b)} = P_{(a,b)old} + (1 - P_{(a,b)old}) \times P_{init}
\]  

(1)

If a pair of nodes does not encounter each other in a while, they are less likely to be good forwarders of messages to each other, thus the delivery predictability values must age, being reduced in the process. The aging equation is shown in Eq. 2, where \( \gamma \in (0,1) \) is the aging constant, and \( k \) is the number of time units that have lapsed since the last time the metric was aged. The time unit used can
differ, and should be defined based on the application and the expected delays in the targeted network.

\[ P(a,b) = P(a,b)_\text{old} \times \gamma^k \quad (2) \]

The delivery predictability also has a transitive property, that is based on the observation that if node A frequently encounters node B, and node B frequently encounters node C, then node C probably is a good node to forward messages destined for node A to. Eq. 3 shows how this transitivity affects the delivery predictability, where \( \beta \in [0, 1] \) is a scaling constant that decides how large impact the transitivity should have on the delivery predictability.

\[ P(a,c) = P(a,c)_\text{old} \times (1 - P(a,c)_\text{old}) \times P(a,b) \times P(b,c) \times \beta \quad (3) \]

A.2 Forwarding strategies

In traditional routing protocols, choosing where to forward a message is usually a simple task; the message is sent to the neighbor that has the path to the destination with the lowest cost (usually the shortest path). Normally the message is also only sent to a single node since the reliability of paths is relatively high. However, in the settings we envision here, things are completely different. For starters, when a message arrives at a node, there might not be a path to the destination available so the node have to buffer the message and upon each encounters with another node, the decision must be made on whether or not to transfer a particular message. It may also be sensible to forward a message to multiple nodes to increase the probability that a message is really delivered to its destination.

Unfortunately, these decisions are not trivial to make. In some cases it might be sensible to select a fixed threshold and only give a message to nodes that have a delivery predictability over that threshold for the destination of the message. On the other hand, when encountering a node with a low delivery predictability, it is not certain that a node with a higher metric will be encountered within reasonable time. Thus, there can also be situations where we might want to be less strict in deciding who to give messages to. Furthermore, there is the problem of deciding how many nodes to give a certain message to. Distributing a message to a large number of nodes will of course increase the probability of delivering a message to its destination, but in return, more system resources will be wasted. On the other hand, giving a message to only a few nodes (maybe even just a single node) will use little system resources, but the probability of delivering a message is probably lower, and the incurred delay high.

In the evaluations in this paper, we have chosen a rather simple forwarding strategy – when two nodes meet, a message is sent to the other node if the delivery predictability of the destination of the message is higher at the other node. The first node does not delete the message after sending it as long as there is sufficient buffer space available (since it might encounter a better node, or even the final destination of the message in the future). If buffers are full when a new message is received, a FIFO system is used to drop the oldest message from the buffer.

IV. Simulations

To evaluate the protocol, we have developed a simple simulator in the Java programming language. The reason for implementing a new simulator instead of using one of the large number of widely available simulators was mainly due to a desire to focus on the operation of the routing protocols instead of simulating the details of the underlying layers that are unlikely to affect the results.

A. Mobility Model

When doing an evaluation of a protocol or system, it is very important that the models used in the evaluation are realistic. Since part of the motivation for this work was the observation that totally random mobility is not likely to be a realistic model, we wanted to design a scenario to reflect this. This resulted in the design of a scenario that we call the “community model”. This scenario consists of a 3000m × 1500m area as shown in Fig. 2. This area is divided into 12 subareas, 11 communities (C1-C11), and one “gathering place” (G). Each node has one home community that it is more likely to visit than other places, and for each community there are a number of nodes that have that as home community. Furthermore, in each community, and at the gathering place, there is a fixed (non-mobile) node as well that could be acting as a gateway for that community. The mobility in this scenario is such that nodes select a destination and a speed, move there, pause there for a while, and select a new destination and speed. The destinations are selected such that if a node is at home, there is a high probability that it will go to the gathering place (but it is also possible for it to go to other places), and if it is away from home, it is very likely that it will return home. Table I shows the probabilities of different destinations being chosen depending on the current location of a node. Real-life scenarios where this kind of mobility can occur include human mobility where the communities are, for example, villages, but also sensor network applications where sensors are attached to animals – in such cases the gathering place may be a feeding ground, and the communities can be herd habitats.

<table>
<thead>
<tr>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
</tr>
</thead>
<tbody>
<tr>
<td>C5</td>
<td>C6</td>
<td>C7</td>
<td>C8</td>
</tr>
<tr>
<td>C9</td>
<td>C10</td>
<td>C11</td>
<td>G</td>
</tr>
</tbody>
</table>

Fig. 2. Community model
The scenario we have used is based on the community mobility model defined above. For each community, there are five nodes that have that as their home community. Nodes select speeds between 10 and 30 m/s after pausing. The traffic in this scenario is also different from the random mobility scenario. Every tenth second, two randomly chosen community gateways generate a message for a gateway at another community or at the gathering place. Five nodes that have that as their home community.

**B. Metrics**

In our evaluation of the two protocols, we have focused on comparing their performance with regard to the following metrics. First of all, we are interested in the *message delivery ability*, i.e., how many of the messages initiated the protocol are able to deliver to the destination. Even though applications using this kind of communication should be relatively delay-tolerant, it is still of interest to consider the *message delivery delay* to find out how long it takes a message to be delivered. Finally, we also study the number of *message exchanges* that occur between nodes. This indicates how the system resource utilization is affected by the different settings, which is crucial so that valuable resources such as bandwidth and energy are not wasted.

**C. Simulation setup**

The scenario we have used is based on the community mobility model defined above. For each community, there are five nodes that have that as their home community. Nodes select speeds between 10 and 30 m/s after pausing. The traffic in this scenario is also different from the random mobility scenario. Every tenth second, two randomly chosen community gateways generate a message for a gateway at another community or at the gathering place. Five seconds after each such message generation, two randomly chosen mobile nodes generate a message to a randomly chosen destination. After 3000 seconds the message generation cease and the simulation is run for another 8000 seconds to allow messages to be delivered. A *warm up* period of 500 seconds is used in the beginning of the simulations before message generation commence, to allow the delivery predictabilities of PROPHET to initialize.

![Fig. 3. Simulation results](image)

### TABLE I

**Destination selection probabilities**

<table>
<thead>
<tr>
<th>From</th>
<th>Home</th>
<th>Gathering place</th>
<th>Elsewhere</th>
</tr>
</thead>
<tbody>
<tr>
<td>Home</td>
<td>0.9</td>
<td>0.8</td>
<td>0.2</td>
</tr>
<tr>
<td>Elsewhere</td>
<td>0.9</td>
<td>-</td>
<td>0.1</td>
</tr>
</tbody>
</table>

### TABLE II

**Parameter settings**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_{init}$</td>
<td>0.75</td>
</tr>
<tr>
<td>$\beta$</td>
<td>0.25</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>0.98</td>
</tr>
</tbody>
</table>

We ran simulations, varying the queue size at the nodes (the number of messages a node can buffer), the communication range of nodes, and the hop count value set in the messages. For each setup, we made 5 simulation runs with different random seed. Table II shows the values for parameters kept fixed in our simulations (initial simulations indicated that those values were reasonable choices for the parameters).

### V. Results

The results presented here are averages from 5 simulation runs, and the error bars in the graphs represent the 95% confidence intervals. For each metric and scenario, there are two graphs with two different values of the hop count setting. Each of these graphs contain curves for both Epidemic Routing and PROPHET for the two different communication ranges. On the x-axis in each graph, the queue size can be found.

First, we investigate the delivery rates of the protocols, as shown in Fig. 3a). It is easy to see that the queue size impacts performance; as the queue size increases, so does...
the number of messages delivered to their destination for both protocols. This is intuitive, since a larger queue size means that more messages can be buffered, and the risk of throwing away a message decreases. A significant difference can be seen between the performance for the two protocols, and it can be seen that PROPHET is at times able to deliver up to twice as many messages as Epidemic Routing. Interesting to note is that the delivery rate (especially for the short communication range) is adversely affected by an increase in the hop count. This is probably due to the fact that with a higher hop count, messages can spread through a larger part of the network, occupying resources that otherwise would be used by other messages, while with a lower hop count, the mobility of the nodes have greater importance.

Looking at the delivery delay graphs in Fig. 3b), it seems like increasing the queue size, also increases the delay for messages. However, the phenomenon seen is probably not mainly that the delay increases for messages that would be delivered even at a smaller queue size (even though large buffers might lead to problems in being able to exchange all messages between two nodes, leading to a higher delay), but the main reason the average delay is higher is coupled to the fact that more messages are delivered. These extra delivered messages are messages that were dropped at smaller queue sizes, but now are able to reside in the queues long enough to be delivered to their destinations. This incurs a longer delay for these messages, increasing the average delay.

Finally, looking at the graphs in Fig. 3c), it can be clearly seen that PROPHET has a lower communication overhead and sends fewer messages than Epidemic routing does. This is due to the fact that when using PROPHET messages are only sent to “better” nodes, while Epidemic routing sends all possible messages to nodes encountered. Another thing that can be seen from the graphs is that increasing the communication range generally increases the performance in terms of delivery rate and delay, but also increases the communication overhead. This is not very surprising, since a larger communication range allows nodes to communicate directly with a larger number of other nodes and increases the probability of two nodes meeting each other.

VI. DISCUSSION AND FUTURE WORK

In our evaluation we have used a FIFO queue at the nodes, so whenever a new message arrives to a full queue, the message that has been in the queue for the longest time is dropped. It might be better to use some other strategy here; for example, dropping the message that has already been forwarded to the largest number of other nodes.

The simple forwarding strategy used by PROPHET in our evaluation worked fairly well, and outperformed Epidemic Routing. Nevertheless, it is still interesting to investigate other forwarding strategies to see if performance can be further enhanced. It can, for example, be beneficial to investigate if it always is a good idea to give messages to nodes with a higher delivery predictability than your-