Probabilistic Routing Protocol for Intermittently Connected Mobile Ad hoc Network* (PROPICMAN)

Hoang Anh Nguyen, Silvia Giordano†, Alessandro Puiatti
ISIN-DTI, SUPSI, Switzerland
[firstname.lastname]@supsi.ch

Abstract

Routing in intermittently connected mobile ad hoc networks (ICMAN) is a very challenging problem because disconnections are prevalent and the lack of knowledge about network dynamics hinders good decision making. In ICMAN one of the most important decisions is how to choose the most suitable intermediate(s) to forward the message to the destination. We propose in this document a new algorithm (PROPICMAN) that, based on context information, allows the sender to select the neighbor(s) such that the message has the highest probability to reach the destination.

1. Introduction

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 best path to the destination (usually the shortest path). Normally the message is also only sent to a single node since the reliability of a path is relatively high. When mobility enters in the game, there are many different scenarios and situations in which communication is inherently intermittent due to the mobility of participants.

In this work, we focus on routing in Intermittently Connected Mobile Ad-hoc Networks (ICMAN) formed by mobile users roaming within small and even large geographical areas, who either have no access to the infrastructure, or, could not or do not want to use the infrastructure for transferring certain delay tolerant data amongst themselves. The type of communication networks addressed in this document is mainly viable for applications that can tolerate long delays and are able to deal with extended disconnection periods.

We find that most of the recent works on this issue did not care so much about the information of mobile devices’ carriers. This reduces the possibilities of considering the users’ social behavior, which is seen as one of the most interesting parameters for mobile users [1,6]. In our approach, the carrier’s information, called node profile, plays an important role in predicting the mobility of the nodes. We consider that people are not likely to move around randomly. Rather, they move in a predictable fashion based on repeating behavioral patterns at different timescales (day, week, month). If a node has visited a place several times before, it is likely that it will visit this location again in the future. For example, because this person likes pasta and the place is an Italian restaurant. Thus, relevant information can be deducted by the node profile. We introduce a new algorithm, called Probabilistic Routing Protocol for Intermittently Connected Mobile Ad hoc Networks (Propicman), which makes use of this observation to improve the accuracy of routing. Based on the node profiles of its two-hop neighbor nodes, a sender can select, as forwarder(s), the node(s) with the highest probability of delivering the message toward the destination (delivery probability). The delivery probability can be considered quasi-static information, hence the mobility of a node does not really affect to the validity of Propicman selection.

Our results show that Propicman exploits effectively the mobility, as well as it reduces significantly the number of nodes involved in the forwarding process. Thus the network overhead is significantly low in comparison with the other recent dissemination-oriented algorithms.

Propicman also takes into account information privacy from design. Most of the solutions proposed to MANET routing force users to share and exchange their information during the routing process [3,4,5], but it is unlikely that people accept that. This is one of

* Partially funded by Haggle project (EU FP6-2004-IST-4)
† IEEE senior member
the main barriers to MANET acceptance [12,13]. In our solution nodes can share their information in a “hidden” format, but this information can still be used for routing. Furthermore, only the destination can read and understand the message content, while this content is hidden from the intermediate nodes.

The rest of the paper is outlined as follows. In section 2, we motivate our work by critically discussing some recent research on routing protocols in this area. In section 3, we introduce Propicman. In section 4, we discuss our security approach. In section 5, we compare Propicman data throughput and network overhead with other solutions. We then present our conclusions in section 6.

2. Related work

Recently, some new routing protocols that exploit nodes mobility have been proposed for MANETs such as Message Ferry[7], CAR[5], Sociological Orbit aware Location Approximation and Routing (SOLAR-HUB)[3,4], Erasure Codes[9], Epidemic Routing[8], Prophet[2], etc. We will shortly introduce the last two, as used in our comparison.

2.1. Prophet

In the Probabilistic Routing Protocol using History of Encounters and Transitivity, the selection of the best neighbor node is based on how frequently a node encounter another. Prophet uses a probabilistic metric called delivery predictability that indicates how likely it is that A will meet B, and thus that will be able to deliver a message to B. When two nodes meet, they exchange their summary vectors, which contain their delivery predictability information. If two nodes do not meet for a while, the delivery predictability reduces. When the sender wants to send a message to the destination D, it will look for the neighbor node that has the highest amount of time encountering D, meaning that has the highest delivery predictability to D [2]. This property is further transitive.

Since Prophet has a similar approach to Propicman, it is a good candidate for analyzing the advantage of context information (see section 5).

2.2. Epidemic routing

Epidemic Routing [8] 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 the destination. Nodes buffer messages when there is no available path to the destination. An index of these messages called a summary vector is kept by the nodes, and when two nodes meet they exchange them. So doing, each node can determine if the other node has some message that it did not see before and requests it. This means that, as long as there is some available buffer spaces, messages will spread epidemically as a disease, as nodes meet and “infect” each other. Besides the obvious fields of source and destination addresses, messages also contain a hop count field. This field 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.

3. System architecture

3.1. Goals and Design Issues of Propicman

The goals of Propicman are to: i) efficiently distribute messages through opportunistic mobile ad hoc networks in a probabilistic fashion, ii) minimize the amount of resources consumed in delivering any single message, and iii) maximize the percentage of messages that are eventually delivered to their destination. To achieve these goals, we use the probability of nodes to meet the destination, and infer from that the delivery probability.

The most innovative aspects of our approach are:
- **Routing with Zero Knowledge**: A sender does not need to have any knowledge of other nodes in the network. It bases its routing decisions only on the information it knows about the destination(s).
- **User profile**: Unlike some other protocols working with profiles, in Propicman, when a node has a message to send to a destination, based on the information about the destination node that the sender knows, it associates the different weights to the attributes of profile (herein called evidences). Thus it can classify which information is important in order to successfully deliver the message to the destination.
- **Security within Community**: Our goal is not to develop a security mechanism that no one can bypass, but to avoid that anyone can easily get some information in the network. We aim to protect not only the message content, but also destination information inside secure communities as in Haggle project [11].

3.2. PROPICMAN

When the node S wants to send a message M to the destination D, it sends to its neighbors M's header. This header contains the information the sender knows about the destination. Based on this information, the
neighbor nodes compute their delivery probability, which we assume are independent from each other. M’s header is forwarded to the two-hop neighbors of S. S will send the message M only on the two-hop route(s) with the highest delivery probability, if this is higher than its own. After sending the message content, S keeps the message for the next eventual encounter.

As also highlighted in [1], two-hop routes are the most suitable choice to exploit more routing information with minimal additional costs and risks. In order to compute the delivery probability at each node, we first define a common set of evidences that is the list of evidences used in the network and the related weight (importance of this evidence in the network). Each node will build its node profile, which is an instantiation of the common set of evidences.

3.2.1 Building a node profile. We reasonably assume that, when the sender wants to send a message, it must have some information (attributes) about the destination node.

We assume that each node has a node profile that contains information about node, such as the device holder information (name, residence address, workplace, nationality, hobbies, etc.), information about device status, such as battery level, memory capacity, etc.

The node profile is represented by the set of evidence/value pairs. To do the mapping evidences that we will discuss in the next section, each evidence/value pair in the node profile is hashed by the hash function $H(E_i, V_i)$ that results in a hashed value with the length of 32.

<table>
<thead>
<tr>
<th>Evidence name (E)</th>
<th>Value (V)</th>
<th>Hashed values</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E_1$</td>
<td>$V_1$</td>
<td>$H(E_1, V_1)$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$E_n$</td>
<td>$V_n$</td>
<td>$H(E_n, V_n)$</td>
</tr>
</tbody>
</table>

### Table 1: Node profile structure ($V_i$ can be empty)

A hashed value of one evidence/value pair has a form of 32 bytes, thus they have the same size as followings: 26a48976c47c0d2b2383a34010eae937

3.2.3. Strategy to select the best Forwarder(s).

Suppose that the source node S - the node holding the message and trying to choose the best route to forward the message M to the destination D - knows some information about D’s profile. From this information (some evidence/value pairs), S builds the message header $H_M$ which is the concatenation of all the hashed pairs of evidences/values as follows:

$$H_M = \text{Concatenation} (H(E_{Di}, V_{Di}), W_i, MAC_S, SN)$$  \hspace{1cm} (1)

Where MAC$_S$ (MAC address of S) and SN (the sequence number of M) are used to avoid duplicated messages, $W_i$ are the weights corresponding to each evidence and are put in the same order as the corresponding evidences.

![Figure 1: A message header](image1.png)

By putting the weights into the message header $H_M$, the sender can “classify” which evidence is important for the intermediate nodes in order to have the chance to meet the destination. For example, if the sender S knows that the destination D is mostly at his office during the day, S will put the evidence “workplace” the highest weight since intermediate nodes that have the same workplace with D will have very high probability to meet D.

At first, S sends the message header $H_M$ to every neighbor node, without including the message content.

![Figure 2: Two-hop route selection](image2.png)

Each neighbor node (for instance, A), when receiving $H_M$, it compares each of its own hashed values of
evidence/value pairs to the hashed values in $H_M$. For each matched element, $A$ gets from $H_M$ the corresponding weight of the matched evidence. From all the elements that matched, $A$ computes its delivery probability to the destination $D$ as follows:

$$0 \leq P_A = \frac{\sum \text{Matched}(W_i)}{\sum W_M} \leq 1 \quad (2)$$

where $\sum \text{Matched}(W_i)$ is the sum of the weights of the evidences in $A$, which hashed values matched the ones in $H_M$, and $\sum W_M$ is the sum of the weights of all the evidences that $S$ knows about $D$ (not empty in $H_M$).

For the next hop level (second hop – figure 2), at its turn, each neighbor of $S$, say $A$, sends $H_M$ to every node in its neighborhood ($A_i$). Every neighbor node of $A$ computes its own delivery probability as in (2).

With all the delivery probabilities of its neighbors, $A$ is able to know the one with highest delivery probability, send the result back to $S$. Then $S$ takes the maximum delivery probability of the two-hop routes, and if this delivery probability is higher than or equal to the delivery probability of $S$. $S$ will send the message content $M$ to the node that has the highest delivery probability through this two-hop route. Thus, on his way to $D$, the message is only forwarded to node(s) with higher delivery probability.

If $S$ wants to ensure a delivery rate or when there are no constraints about the network overhead, $S$ can choose more than one (highest probability) route, then it replicates and sends $M$ over those routes. This is a trade-off between data latency and network overhead. If the number of selected routes equals to 1, we have the lowest overhead, but quite long delay. This is further analyzed in section 5.

After sending the message content, the intermediate nodes still keep the message for the next encounters.

4. Security considerations

One of the most important issues in ad hoc networks is security. In [3,4,5,6], members in ad hoc networks can be grouped in different communities. Here we assume that security measures, such as authentication, signature, etc., are implemented at community level.

In our proposal, we aim to ensure message privacy and member’s information within a community. When a node $S$ wants to send a message $M$ to another node $D$, it can happen that $S$ uses some intermediate nodes. In Propicman, before sending the message content $M$, $S$ sends $H_M$ that carries some information about $D$ (evidences), and this could be captured. However, we hide the plain text of $H_M$ (and the message content $M$) to the intermediate nodes. Only the destination node $D$ can recover and read $M$. The intermediate nodes are not allowed to know to whom this message is sent, and the content of the message. As mentioned in the subsection 3.1, in this paper we are not aiming to build a complete security mechanism that no one can bypass, this solution is considered as a suggestion to use among members inside a community.

4.1. Hiding destination’s information:

We adopt a one-way hash function $H$ that uses the message digest algorithm to hash the evidences of the destination node in $H_M$.

As described in the section 3.2, the profiles of nodes also contain the hashed values $H(E_i, V_i)$. Every time a node receives $H_M$, it compares all its hashed values with the hashed values in $H_M$. Thus, it can only “spy” some information of $D$ in $H_M$ if it matches its hashed values. But, it cannot know about the other non-matched pairs in $H_M$. And $D$ is the only one who can get the complete information, as it is the only one who can match all the hashed values in $H_M$ with the ones in its profile.

4.2. Encrypting/Decrypting message content:

We also hide the message content from “unintended” nodes. Only the destination node can receive and understand (recover) the message content. As we discussed above, the information of the destination node is hidden from others by using the function $H$. Thus, we can make use of this fact to encrypt/decrypt the message content.

The encryption scheme consists of an encryption operation and a decryption operation, where the encryption operation produces a cipher text from the message content with some keys, and the decryption operation recovers the message from the cipher text under the same keys.

$$F_{enc}(K, M),$$

where $M$ is the message content and $K$ is the encryption key.

We use the information (evidences/values) that the sender knows about the destination node as keys. With this technique, in a community scenario, only the destination node can decrypt the cipher message to get the plain message since it knows the keys.

$$K = \text{Concatenation}(E_iV_i)_{i=1}^n$$

where $n$ is the number of evidences. $V_i$ will be empty if $S$ does not know about $E_i$ of $D$. 

5. Simulation and Performance evaluation

5.1. Methodology and Implementation:

We suppose that there are no battery consumption or network bandwidth constraints in the simulation. With these assumptions, we compare the routing performance of Propicman with Prophet and Dynamic Epidemic using the following two metrics:

- **Network Byte Overhead:** the ratio of the amount of control and data information (measured in bytes) that got transmitted from a node to its neighbors, over the total amount of data information (measured in bytes) that was received correctly, for the entire duration of the simulation.

- **Average End-to-End Data Delay:** the end-to-end data delay is defined as the time interval (measured in seconds) between the generation of a data packet at the source and its reception at the destination. This value is averaged over all packets correctly received at the destination.

The simulation focuses on the operation of routing protocol and does not simulate the details of the underlying layers.

From investigations of daily activities in our department, the scenario is designed as a rectangular of 870x520 meters, divided into 6 areas. Each person (node) is randomly distributed across the 6 areas, with different weights for each area, depending on the person. The nodes will probably be, for most of the (daily) time, in the working room (highest probability), then in the canteen and so on. For example, node 1 will be 89% in his working room, 6% in the canteen, 3% in the rest room, 2% at the coffee machine area, and 0% in the other spaces. We did not include a specific mobility model, because, as mentioned above, the delivery probability is quasi-static information. We assume that nodes have the same cover range, and move from one area to another based on the probabilistic distribution.

All the parameters of Prophet we use in the simulation are the same as in [2].

5.2. Simulation results and analysis:

Our system relies on predictions about the future values of context evidences. However, predictions are not always reliable because of the human-based behavior exists in the prediction process. The result presented in the paper is the one we take with the standard confidence level of 95% and the confidence interval at each test is [X-970, X+970] (msec).

We ran our probabilistic algorithm against the Prophet and Dynamic Epidemic algorithms. We evaluated the transmission delay and the network overhead of each algorithm in two different network sizes: 30 nodes and 50 nodes. We sort the results for clearly making the comparison between different approaches.

As we have no bandwidth constraints, the Epidemic algorithm clearly will have the highest performance for the transmission delay. The sender in the Epidemic algorithm will “flood” all the nodes that it meets on the way of movement, and so on. This is illustrated in figure 3 and figure 5.

Prophet uses the encounter history of nodes to predict the meeting probability between nodes. It supposes that if two nodes meet each other very often in the past, they likely meet again in the future. But in real life, this is not always correct, and needs further refinement. For this reason, we introduced the priority of information (evidences) of nodes. By putting different weights to the evidences, the selection of candidates among the neighbor nodes results more specific and accurate.

As clear, the Epidemic routing algorithm performs very badly for network overhead. Furthermore, in a network with high mobility, the flooding load increases with mobility. Prophet and Propicman only forward the message to the neighbor node that has the higher delivery probability, thus are no affected by mobility.

![Figure 3: Delay (30 nodes)](image-url)
This technique reduces significantly the network overhead. Each source node in Propicman adds $32n + 19$ bytes of traffic in the network where $n$ is the number of evidences in the common profile, 19 bytes are the size of MAC address of $S$ and the sequence number of $M$. Meanwhile, in Prophet, when two any nodes meet, they exchange MAC address and summary vectors. We can see that the analysis results on the network overhead are also similar to the simulation results in the both cases Prophet and Propicman.

6. Conclusions

In this paper, we have proposed Propicman, a probabilistic protocol that uses the node profiles and information weights to select the route with highest delivery probability. Simulations have shown that in a community based scenario, Propicman clearly gives better performance than similar approaches as Epidemic Routing and Prophet. We are currently evaluating more complex scenarios that include more nodes, larger profiles and larger areas. We are also considering mechanisms to learn from the past to adjust the weights of evidences, and studying the overhead of a more proactive approach, where nodes distribute the profiles for two hops, and thus the highest delivery probability route computation can be done at the sender. Finally, we are setting up real experiments to see the performance in realistic scenarios.

7. Acknowledgements

We want to thank our colleague Prof. Marco Zaffalon, for his valuable suggestions in the graph analysis.

8. References