Anchored Path Discovery in Terminode Routing

Ljubica Blažević, Silvia Giordano, and Jean-Yves Le Boudec

Laboratory for computer Communications and Applications (LCA)
Swiss Federal Institute of Technology, Lausanne (EPFL), Switzerland
{ljubica.blazevic, silvia.giordano,jean-yves.leboudec}@epfl.ch

Abstract. Terminode routing, defined for potentially very large mobile ad hoc networks, forwards packets along anchored paths. An anchored path is a list of fixed geographic points, called anchors. Given that geographic points do not move, the advantage to traditional routing paths is that an anchored path is always "valid". In order to forward packets along anchored paths, the source needs to acquire them by means of path discovery methods. We present two of such methods: Friend Assisted Path Discovery assumes a common protocol in all nodes and a high collaboration among nodes for providing paths. It is a social oriented path discovery scheme. Geographic Maps-based Path Discovery needs to have or to build a summarized view of the network topology, but does not require explicit collaboration of nodes for acquiring path. The two schemes are complementary and can coexist.

1 Introduction

Routing in mobile ad hoc networks (Manets [7]) is already a difficult task when the network size is considerably small, as studied in most of the Manets' protocols. When the network size increases, the routing task becomes too hard to be addressed with traditional approaches. We consider a large mobile ad hoc network, referred to as *terminode* network. Each node, called terminode here, has a permanent End-system Unique Identifier (EUI), and a temporary, location-dependent address (LDA).

Terminode routing [2], which was proposed for coping with this scenario, is a combination of two routing protocols: *Terminode Local Routing (TLR)* and *Terminode Remote Routing (TRR)*. TLR is a mechanism that allows for destinations to be reached in the vicinity of a terminode and does not use location information for taking packet forwarding decisions. It uses local routing tables that every terminode proactively maintains for its close terminodes. In contrast, TRR is used to send data to remote destinations and uses geographic information; it is the key element for achieving scalability and reduced dependence on intermediate systems. TRR default method is *Geodesic Packet Forwarding (GPF)*. GPF is basically a greedy method that forwards the packet closer to the destination location until the destination is reached. GPF does not perform well if the source and the destination are not well connected along the shortest geodesic path. If the source estimates that GPF cannot successfully reach the destination, it uses *anchored paths*. In contrast with traditional routing algorithms, an anchored path does not consist of a list of nodes to be visited for reaching the destination. An anchored path is a list of fixed geographic points, called anchors. In traditional paths made of lists of nodes, if nodes move far from where they were at the time when the path was computed, the path cannot be used to reach the destination. Given that geographic points do not move, the advantage of anchored paths is that an anchored path is always "valid". In order to forward packets along an anchored path, TRR uses the method called *Anchored Geodesic Packet Forwarding (AGPF)*, described in [2]. AGPF is a loose source routing method designed to be robust for mobile networks. A source terminode adds to the packet a route vector made of a list of anchors, which is used as loose source routing information. Between anchors, geodesic packet forwarding is employed. When a relaying terminode receives a packet with a route vector, it checks whether it is close to the first anchor in the list. If so, it removes the first anchor and sends the packet towards the next anchor or the final destination using geodesic packet forwarding. If the anchors are correctly set, then the packet will arrive at the destination with a high probability. Simulation results show that the introduction of the anchored paths is beneficial or the packet delivery rate [2].

In order to forward packets along anchored paths, the source needs to acquire them by means of path discovery methods. We presented in [2,1] the basic concepts of two such methods: *Friend Assisted Path Discovery (FAPD)*, and *Geographic Maps-based Path Discovery (GMPD)*. FAPD enables the source to learn the anchored path(s) to the destination using, so-called, $friends$, terminodes where the source already knows how to route packets. We describe how nodes select their lists of friends and how these lists are maintained. GMPD assumes that all nodes in the network have a complete or partial knowledge of the network topology. We assume that nodes are always collaborative, that they do not behave maliciously and that they perform protocol actions, whenever requested, in the appropriate way. In this paper we describe FAPD and GMPD.

2 Friend Assisted Path Discovery

FAPD is a default protocol for obtaining anchored paths. It is based on the concept of small-world graphs (SWG) [9]. SWG are very large graphs that tend to be sparse, clustered, and have a small diameter. The small-world phenomenon was inaugurated as an area of experimental study in social science through the work of Stanley Milgram in the 60's. These experiments have shown that the acquaintanceship graph connecting the entire human population has a diameter of six or less; this phenomenon allows people to speak of the "six-degrees of separation". We view a terminode network as a large graph, with edges representing the "friend relationship". B is a $friend$ of A if (1) A thinks that it has a good path to B and (2) A decides to keep B in its list of friends. A may have a good path to B because A can reach B by applying TLR, or by geodesic packet forwarding, or because A managed to maintain one or several anchored paths to B that work well. The value of a path is given in terms of congestion feedback information such as packet loss and delay. Path evaluation is out of the goals of this paper. By means of the TLR protocol, every terminode has knowledge of a number of close terminodes; this makes a graph highly clustered. In addition, every terminode has a number of remote friends to which it maintains a good path(s). We conjecture that this graph has the properties of a SWG. That is, roughly speaking, any two vertices are likely to be connected through a short sequence of intermediate vertices. This means that any two terminodes are likely to be connected with a small number of intermediate friends. With FADP, each terminode keeps the list of its friends with the following information: location of friend, path(s) to friend and potentially some information about the quality of path(s). FAPD is composed by two elements: *Friends Assisted Path Discovery Protocol (FAPDP)* and *Friends Management (FM).*

2.1 Friend Assisted Path Discovery Protocol (FAPDP)

FAPDP is a distributed method for finding an anchored path between two terminodes in a terminode network. When a source S wants to discover a path to destination D, it requests assistance from some friend. If this friend is in condition to collaborate, it tries to provide S with some path to D (it can have it already or try to find it, perhaps with the collaboration of its own friends). Figures 1 and 2 present FAPDP in pseudocode at the source and at an intermediate friend.

```
if (S has a friend F1 where dist(F1,D)<dist(S,D) )
   {S sets "F" bit in the packet header; send a packet to F1;}
else if (S has a friend F3 such that dist(S, F3) < max_dist )
     {S sets "F" bit in the packet header;
        tabu_index=1; min_dist=dist(S,D); //start tabu mode
           send the packet to F3;}
else apply geodesic packet forwarding (GPF) to D;
```

Fig. 1. Friend Assisted Path Discovery Protocol at the source

When source S, which has some data to send to D, has some friends that are closer to D than S itself, it selects friend $F1$ whose location is closest to D, and starts FAPDP with $F1$. S sends the data packet to $F1$ according to the existing path that S maintains to $F1$ because $F1$ is a friend of S. S sets, within the data packet header, the "F" bit[1]. This denotes that the corresponding packet is a *path discovery packet (PDP).*

The *fapd_anchored_path* field inside the path discovery packet progressively contains anchor points from S to D.

If S has an anchored path to $F1$, S simply puts anchors of this path in *fapd_anchored_path* field (S sends data to $F1$ with AGPF). Otherwise, S leaves this field empty (in this case S sends to $F1$ with geodesic packet forwarding). Upon reception of the path discovery packet, $F1$ puts its geographic location (LDA_{F1}) inside *fapd_anchored_path* field as one anchor. If $F1$ has an anchored path to D, $F1$ appends this path into *fapd_anchored_path* field and sends the packet to D by AGPF. If $F1$ does not have a path to D, it recursively uses FAPDP: it checks if it has a friend $F2$ closer to D, and then it performs the same steps as S. This is repeated until the packet is received by some intermediate node that finds D can be reached by means of TLR and it forwards the packet to D by TLR.

[1] the "F" bit is not reset before reaching D

```
F1 is intended receiver of a path discovery packet ("F"bit = 1 ): S needs a path to D
if (F1 == D) {send path reply with fapd_anchored_path to S;}
else if (F1 has a path to D)
  append this path in fapd_anchored_path and send the packet to D;
else if (tabu_index > 0 ) //packet in tabu mode
   {
    if ( F1 has a friend F2 where dist(F2, D) < min_dist)
      {tabu_index=0; send the packet to F2}
    else if (tabu_index < 2 and F1 has a friend F3 such that dist(F1, F3) < max_dist )
         { tabu_index++; send a packet to F3}
    else // tabu_index reached the maximum value
        {send a packet to D by geodesic packet forwarding}
   }
else //packet not in tabu mode
 {
  if (F1 has a friend F2 where dist(F2,D)<dist(F1,D) )
   send a packet to F2;
  else if (F1 has a friend F3 such that dist(F1, F3) < max_dist )
      {tabu_index=1; min_dist=dist(F1,D); send a packet to F3}// start tabu mode
  else apply geodesic packet forwarding (GPF) to D;
}
```

Fig. 2. FAPDP at the intermediate friend and at the destination

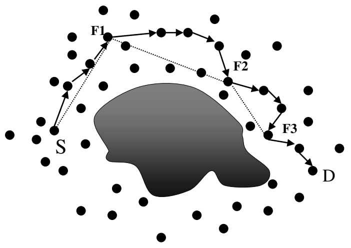

Fig. 3. Figure presents how FAPDP works when source S, has a friend $F1$ that is closer to D than S. S sends data packet to $F1$ and sets the "F" bit in the packet header in order to denote that this is a "path discovery packet". Upon reception of the path discovery packet PDP, $F1$ puts LDA_{F1} inside the $fapd\_anchored\_path$ field of PDP as one anchor. In this example $F1$ does not have path to D, but has a friend $F2$ whose distance to D is smaller than the distance from $F1$ to D. $F1$ sends a PDP to $F2$. In a similar way, $F2$ sends the PDP to its friend $F3$. Once $F3$ receives the PDP, it finds out that D is TLR-reachable and $F3$ forwards the PDP to D by TLR. When D receives the PDP with set "F" bit, it should send back to S a "path reply" control packet with the acquired anchored path from S to D. Assuming that the path from S to $F1$, from $F1$ to $F2$ and from $F2$ to $F3$ does not contain any anchors, the anchored path from S to D is thus a list of anchors $(LDA_{F1}, LDA_{F2}, LDA_{F3})$.

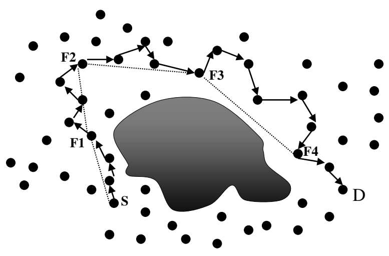

Fig. 4. Figure presents how FAPDP works when source S does not have a friend that is closer to D than itself. S contacts its friend $F1$ that is farther from D in geometrical distance than S is, but such that $dist(S, F1) < max\_dist$. As in the previous example, S sends data packet to $F1$ with "F" bit set. In addition S sets the $tabu\_index$ field to 1 and thus starts the tabu mode of FAPDP. S puts $dist(S, D)$ within min_dist field. Upon reception of the path discovery packet PDP, $F1$ finds out that it does not have a friend whose distance to D is smaller than min_dist. $F1$ forwards the PDP to its friend $F2$ (that is in the opposite direction from D) where $dist(F1, F2) < max\_dist$, and sets $tabu\_index$ to 2. Upon reception of the PDP, $F2$ checks that $tabu\_index$ is equal to its maximum value, and $F2$ cannot forward the PDP to its friend that does not reduce the distance min_dist. In our example, $F2$ has a friend $F3$ whose distance to D is smaller than min_dist and forwards the PDP to it. At $F3$, $tabu\_index$ is reset to 0. This means that FAPDP is not longer in tabu mode. From $F3$ the PDP is forwarded to its friend $F4$ and from there to D by using the TLR protocol. The anchored path from S to D is thus a list of anchors $(LDA_{F1}, LDA_{F2}, LDA_{F3}, LDA_{F4})$

However, there are situations where the source or an intermediate friend does not have a friend closer to the destination. For example, in topologies with obstacles, at some point, going in the direction opposite from the destination may be the only way to reach the destination. Therefore, FAPDP permits to T (the source or an intermediate friend) to send a path discovery packet to a friend even though the packet is not getting closer to the destination. However, such a friend must not be distant from T more than distance max_dist[2]. At that point, it starts the "tabu" mode of FAPDP. When in tabu mode, the packet can be sent in a direction opposite to D for a limited number of times. This is inspired to the Tabu Search heuristic ([4], [5]). Tabu Search can be defined as a general heuristic in which a local search procedure is applied at each step of the general iterative process. It could be superimposed on other heuristics to prevent those being trapped in a local minimum. We use the tabu mechanism in order to get out of a local minimum that can happen at some node that does not have a friend closer to the destination. With the tabu mechanism, we try the opposite direction from the destination with the aim to finally get out of a local minimum and further approach towards the destination. Tabu mode is denoted at T by setting the $tabu\_index$ field inside the packet to 1 (default value of $tabu\_index$ is 0). Tabu mode mechanism uses a field called min_dist, where the terminode that started the tabu mode puts its distance to the destination. When an intermediate friend $F1$ receives the path discovery packet, which is in tabu mode, it first checks if it has a friend whose distance to D is smaller than min_dist. If this is the case, the packet is sent to such a friend, and $tabu\_index$ is reset to 0. Otherwise, $F1$ may

[2] we use max_dist equal to five times the transmission range of a terminode

forward the packet to its friend $F2$ whose distance to D is more than min_dist and $F2$ increments $tabu\_index$. In FAPDP, the number of times that the packet is forwarded to a friend that is further from D than min_dist is limited to two (i.e., the value of $tabu\_index$ must not be larger than two). Tabu mode mechanism stops either because a friend that is a distance from D less than min_dist is found, or because $tabu\_index$ is equal to 2. In the second case the packet is forwarded directly to D by geodesic packet forwarding.

Finally, when D receives the packet with the “F" bit equal to one, D must send back to S a *path reply* control packet with the acquired anchored path from S to D. This packet is sent to S by reverting the anchored path and applying AGPF. Once S receives from D a packet with the anchored path, S stores this path in its route cache. If S does not receive an anchored path within some time, or if S wants more paths to D, S starts FAPDP with some other friend. The example presented in Figure 3 shows the case where the path from S to D is found by using three intermediate friends. The example in Figure 4 illustrates the tabu mode of FAPDP.

2.2 Friends Management (FM)

$FriendsManagement(FM)$ is the set of procedures for selecting, monitoring and evaluating friends. For each node, FM maintains a (fixed-size) set of nodes: the *list of friends*. The list of friends contains the nodes that are contacted with FAPDP for discovering paths. Friends Management consists of the following components: *Friends Monitoring*, *Friends Evaluation, Potential Friends Discovery* and *Friends Selection*. FM is critical in the initial phase (bootstrapping). When a node bootstraps, it does not have any information on (possible) friends. Then, the Potential Friends Discovery component is invoked by a node, with the aim of learning, from other nodes, information about some potential friends. Potential friends are also subject to the Friends Selection action. A number of friends are selected from the list of potential friends, taking into account their geographic positions in order to build a friendship graph with the small world graph properties.

Friends Monitoring and Friends Evaluation. Friends are periodically evaluated in order to assure the consistency of the information on current friends and for testing the validity of these friends. We assume that some form of location tracking is active between friends. The *Friends Monitoring* component of FM keeps under control, for a node A, a set of parameters for each friend F_i of A. This consists of:

- Value of the path(s) to the friend F_i: A may evaluate that the path(s) to its friend F_i, that worked well in the past, deteriorated.
- Location of the friend F_i and the average distance to the friend F_i: F_i may have moved considerably from the location where it was at the time when it was included in A's list of friends.
- The number of times friend F_i was contacted to provide a path and the number of paths that are found with the help of the friend F_i: A may contact F_i in FAPDP to learn the path to a given destination, but the path does not come back to A.

Based on these parameters, the *Friends Evaluation* component periodically evaluates whether it is beneficial to keep a node in the list of friends, or if it is better to discard it. Friends with bad evaluation results are discarded from the friends list. At run-time, initial friends disappear- very likely, in order to be substituted by more valid friends. If the number of friends that remain in a list after evaluation is low, new $potential friends$ can be obtained with the Potential Friends Discovery component.

Potential Friends Discovery. Terminode T can have frequent communications with some other terminodes (e.g., for personal, social, business, and economical interest). These terminodes can be directly selected as friends, because it is of T 's interest to maintain constantly a path to them. However, in general, a node could have a small number of terminodes that it contacts frequently. Therefore, it is necessary an automatic mechanism to select new friends. This is the task of the *Potential Friends Discovery (PFD)* and the *Friends Selection (FS)* components. With the PFD component, T receives information on some possible friends from other nodes in the network. This applies both at the bootstrap phase, and periodically, under request from the friend management component. Terminodes periodically send HELLO messages, for the purpose of building of the TLR routing tables [2]. In this process, can learn about the EUIs and the LDAs of the one-hop and the two-hops distant nodes. Given that this information is periodically maintained, a node has information about close nodes at all time, which can be been considered as *close potential friends*. Potential friends that are further than two hops (i.e. T does not maintain information about their EUIs and the LDAs by means of HELLO messages) are called *remote potential friends*. One way for T to learn about remote potential friends is to extract this information from its previous communications. However, to avoid situations where this implicit discovery would not perform properly (e.g. after a long IDLE or OFF period), this component includes a protocol that enables a node to explicitly discover remote potential friends. In this scheme, each node T sends the $get\_friends\_request$ message towards four geographic points ($GP1$, $GP2$, $GP3$ and $GP4$). These points are randomly selected as four points in orthogonal directions at four times the transmission range of T. Once node Y on the way towards a point GP_i finds that GP_i is reachable with TLR, it stops forwarding the message. Then Y sends back the $get\_friends\_reply$ message to T, which contains the list of friends of Y. If this table is empty, Y puts itself in the content field of the message. When node T eventually receives the $get\_friends\_reply$ message from node Y, it combines the received information with the current one of its potential friends. In the case T does not receive a sufficient number of potential friends as reply, it selects four new orthogonal geographic points and repeats the steps described above. After T acquires a list of potential friends, it applies the FS to select a certain number of friends[3] that it includes in the friends list.

Friends Selection: On How to Build a Small World Graph of Friends. The goal of *Friends Selection* component is that terminodes select their friends in a way such that the resulting friendship graph has the properties of a small world graph, as assumed by

[3] We do not define, in this context, how large is the number of friend that a node maintains in its list. This is matter of ongoing work.

FAPD. In the friendship graph vertices correspond to terminodes and there is the edge between nodes i and j, if node i has as a friend node j. The key to generate the small-world phenomenon is the presence of a small fraction of long-range edges, which connect otherwise distant parts of the graph, while most edges remain local, thus contributing to the high clustering property of the graph. Our strategy is to consider geographic positions of nodes when building friendship connections. We distinguish two types of friendship connections:

– short-range friendship connections (local) correspond to connections to one hop distant terminodes (physical neighbours). These local friendship connections aim to make a friendship graph clustered.
– long-range friendship connections (shortcuts): correspond to “logical" connections to terminodes that are more than one hop distant. Each node chooses a small number of them. A shortcut is represented in a friendship graph as one edge.

Our strategy for choosing long-range contacts is inspired by Kleingberg's paper [6]. In order to determine its shortcuts, a node takes into consideration the distances from the other nodes in the graph. A node chooses with higher probability friends that are closer to it. However, there is always some probability that it will choose some distant friend. Kleingberg considers a two-dimensional square lattice, where each node is joined to its four nearest neighbours. Then, for each vertex one shortcut is added, but not purely at random. For each vertex, all the possible destinations of a shortcut link are assigned a rank based on their lattice distance from the source vertex. The probability of choosing a vertex at distance d is proportional to d^{-r}, where r is an additional parameter of the model. In the case when $r = 0$, shortcuts are chosen with uniform probability. Then with a high probability, there are paths between every pair of nodes and these paths are bounded by a polynomial in $log(n)$, exponentially smaller than the total number of nodes n. However, there is no way for a decentralized algorithm to find these paths. When r is large, then only close nodes have a chance to be connected with a shortcut. The key value for r turns out to be 2 [6]. When $r = 2$, it is shown that the resulting graph is a small world graph and there is a distributed algorithm for finding short paths between any two vertices (paths are exponentially smaller than the total number of nodes). This algorithm is greedy, described as follows in Kleinberg's paper: in order to find a path from vertex S to vertex D, S lists all edges that come out of it, and chooses the one that connects S to the vertex that is closest to D, as measured by lattice distance; then repeat the same procedure until D is reached.

Inspired by the Kleinberg's results, we propose the following. The probability that node X_i selects node X_j as its long-distance friend from the list of potential friends $(X_k, k \in 1..n, k \neq i)$ is given with the formula:

$$p(X_i, X_j) = \frac{1/dist^2(X_i, X_j)}{\sum_{k=1,k\neq i}^{n} 1/dist^2(X_i, X_k)} \tag{1}$$

In this formula the probability for node X_i of choosing a friend X_j is proportional to $d(X_i, X_j)^{-2}$. Long-range friend connections are thus selected at random and are not necessarily bi-directional. X_i may have X_j as a friend, but X_j may not have X_i as a friend.

We used simulations to verify our strategy in selecting long-range friendship connections. Simulations were performed with the following assumptions:

– Nodes in the network are distributed as a two-dimensional Poisson point process with a given density.
– All nodes have the same the transmission range R.
– All nodes have a knowledge of identities and locations of all other nodes.

Initially, a friendship graph contains only short-range connections. There is a short-range connection between X_i and X_j, if dist(X_i,X_j)$\leq R$. Then, every node selects a number of shortcuts from its list of potential friends. We performed simulations where this number is equal to one or two. In our simulations, a node has in a list of potential friends the whole set of nodes except nodes with whom short-range connections are already established.

The algorithm that node X_i uses to select its friends consists of three steps:

– Step 1: If node X_i keeps n nodes in its list of potential friends, interval $[0, 1]$ is divided into n intervals. The length of j^{th} interval is equal to $p(X_i, X_j)$, given by Equation (1).
– Step 2: For each friend to be selected, a random trial is performed: a uniform random deviate r in interval $[0, 1]$ is generated. If r falls in the j^{th} interval, X_j becomes a friend of X_i.
– Step 3: The same procedure is repeated for each friend that X_i selects.

The friendship graph is built when all nodes select their friends. Then, we find the *characteristic path length (CPL)* of a graph is the median of the means of the shortest path length connecting each vertex to all other vertices. CPL in a way presents the typical shortest path length between every vertex and every other vertex. CPL is also used by Watts in [9] as a metric to verify whether a graph is a small world graph, with thus a small CPL.

With simulations we want to verify that adding a small number of shortcuts in a friendship graph reduces the CPL of the graph, and that a greedy algorithm for finding paths succeeds in finding short paths. With the greedy algorithm, the source and every intermediate node forward the packet to their short or long-range friend that is closest to the destination. FAPDP is basically a greedy algorithm, and uses the “tabu" mode of operation only when the greedy forwarding is not possible. In our simulation we also calculate CPL, where instead of shortest paths between nodes, we use paths lengths obtained by the greedy algorithm.

Simulation results are given in Figure 5, averaged over ten realizations of random graphs for a given number of nodes. In our simulations, transmission range is (R) is 250 meters. Node density is such that every node has an average of ten neighbours (short-range friendship connections). As we increase the number of nodes, we increase the simulation area, but we keep the same density of nodes. The chosen density ensures that the greedy algorithm succeeds in finding most of the paths. We verified that for less than 5% of pairs of nodes, the greedy algorithm is not able to find a path.

Our simulations have shown the following. First, CPL of the friendship graph exhibits logarithmic length scaling with respect to number of nodes in the graph (see Figure 5).

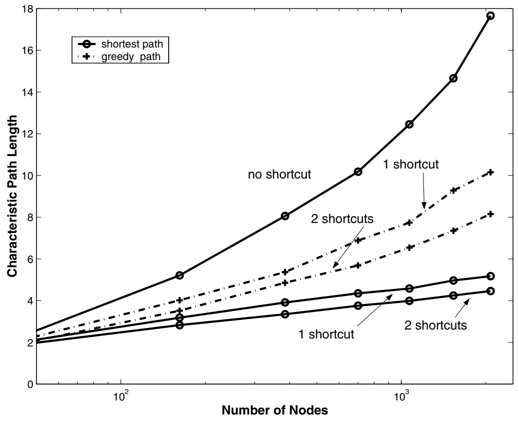

Fig. 5. Figure presents the characteristic path length (CPL) of friendship graph for different number of nodes. Adding a small number of long-range shortcut edges in the graph reduces CPL.

This is a property of a small world graph. A small number of long-range connections (e.g., 1 or 2) are enough to reduce CPL considerably from the case when shortcuts are not used. Second, the greedy algorithm for finding paths succeeds in finding paths whose length is close to shortest paths.

3 Geographic Map-Based Path Discovery (GMPD)

We believe that a good model of a large mobile network does not assume that nodes are uniformly distributed in the network. In order to model a terminode network, we identify the areas with a higher node density, which we call $towns$. Two towns are interconnected by all the nodes in between them (we call it a $highway$). If two towns are interconnected with a highway, there is a high probability that there are terminodes to ensure connectivity from one town to another. GMPD assumes that each terminode has such a summarized geographic view of the network: each terminode has a knowledge of the map of towns. This map defines the town areas and reports the existence of highways between towns. As a first attempt, we model a town area as a square centered in a geographic centre. For each town, the map gives the position of its centre and the size of the square area. The map of a network can be presented as a graph with nodes corresponding to towns and edges corresponding to highways, see Figure 6. Macroscopically, the graph of towns does not change frequently.

GMPD with a given map of towns works as follows:

- Source S determines from its own location (LDA_S) the town area (ST) in which S is situated (or, the nearest town to LDA_S if it is not in the town area). In addition, since S knows the position of destination D (LDA_D), it can determine the town area DT where D is situated (or, the nearest town to LDA_D if it is not in the town area).

- Then, S accesses the network map in order to find the anchored path from S to D. We call this operation a *map lookup*. An anchored path is the list of the geographical points: the points correspond to centers of the towns that the packet has to visit from ST in order to reach DT. One possible realization of the map lookup operation is to find a list of towns that are on the shortest path from ST to DT in the graph of towns; the length of a path can be given either as the number of towns between ST and DT, or the length of the topological (Euclidean) shortest path connecting ST and DT in a graph of towns.

3.1 GMPD with No Initial Summarized View of the Network

Here, we still assume the model of a network based on towns and highways, however, nodes in the network have a constrained view of a network.

A terminode initially does not have the knowledge of a map of towns. The information that a terminode has is the following: 1) if a terminode is within a town area, it knows about neighboring towns areas and town centers with which its current town is connected by highways; 2) if a terminode is on the highway, it knows towns that this highway connects. We assume that a graph of towns is planar. As above, we assume that there is a mapping between the location of a terminode and the corresponding town. Thus, source S can determine from LDA_S the town area (ST) in which S is situated (or, the nearest town to LDA_S if it is not in the town area). In addition, S determines from the LDA_D the town area DT where D is situated (or, the nearest town to LDA_D if it is not in the town area).

Here we present the description of the path discovery algorithm with these assumptions. The source is in town ST and the destination is in town DT.

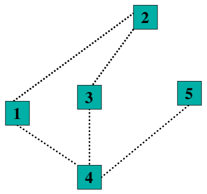

Fig. 6. Figure presents one example of a map of a terminode network. Five town areas (1, 2, 3, 4 and 5) are presented with shaded squares. A highway between two towns is presented with a line between two town areas.

- if ST and DT are the same, then S does not perform path discovery; S sends the packet to D by GPF. If ST and DT are not the same, S begins anchored path discovery. S uses a greedy method: it sends the path discovery packet towards a neighboring town NT whose center is closest to DT. S sends the path discovery packet by using GPF towards the center of NT. As soon as the path discovery packet is received by some terminode N in NT, N adds the center of NT as one anchor in the accumulated anchored path. In addition, N adds NT in the accumulated list of towns (the list of towns is used to record towns that the path discovery packet has visited. This list is later used to simplify the path from ST to DT). If N has a path to DT, it adds this path to the the accumulated anchored path and applies this path to send the packet to D. Otherwise, N repeats the same steps as S.

– if S or an intermediate node (that has to determine the next town to which to send the path discovery packet) does not find a neighboring town closer to DT, the path discovery packet is sent in $perimeter$ mode. In this case, a planar graph traversal method [3] is used, as explained in the example below. This means that next town to send the packet to is determined by using planar graph traversal method applied on a graph of towns.
  As soon as the path discovery packet arrives at some town that is closer to DT than the town where the planar graph traversal method is started, the greedy method resumes in order to find the next town to send the path discovery.
– the explained procedure is repeated until the packet is received by some node in DT. Then the packet is sent directly to D by GPF. When D receives the packet, it analyzes the path and filters possible loops. Then the path is sent back S.
– when S receives the path to DT it adds this path in the list of anchored paths.

We illustrate the path discovery with a localized view of the network with one example. Assume in Figure 6 that source S is in town 1, and destination D is in town 5. S chooses to send the path discovery packet towards the center of town 2 because the center of town 2 is a neighboring town closest to town 5. The first node in town 2 that receives the path discovery packet puts the center of town 2 as one anchor in the accumulated anchored path. Now, because the packet cannot be forwarded closer to town 5, the planar graph traversal algorithm is used. This algorithm is applied on the planar graph of the network map. Following the right hand rule, the first edge of the graph in the direction counterclockwise from the line connecting town 2 and town 5 is the edge that connects town 2 to town 1. Therefore, the packet is sent again to town 1. The planar graph traversal is continued and from town 1 the packet is forwarded to town 4, and then to town 5. As soon as some terminode in town 5 receives the packet it sends it directly to D by using GPF. When D receives the path discovery packet, the accumulated town list is $\{$1,2,1,4,5$\}$. D simplifies the list such that the same town is not visited more than once. The anchored path to be returned to S is the list of towns $\{$1,4,5$\}$ centers.

4 Conclusion

Routing based on anchored paths (AGPF) is a scheme proposed for routing in very large mobile ad hoc networks. Simulation results shows that AGPF is beneficial when there are holes in terminodes distribution and the source cannot reach the destination over the direct geodesic path [2]. This paper presents two schemes for discovering anchored paths, which completes the whole picture. These two schemes, Friend Assisted Path Discovery (FAPD) and Geographic Maps-based Path Discovery (GMPD), are based on two complementary approaches: the social one, e.g. the small world graph approach and the topological one, based on a summarized view of the network. With the most suitable of the two scheme, or with both, a source is able to discover an anchored path to any destination, and thus use the AGPF. We also demonstrated that the resulting friendship graph obtained with FAPD has the wished properties of a small world graph, and illustrated, by examples, how the schemes work. The implementation in the GloMoSim simulator [8] and the further simulation are matter of ongoing work.

References

1. L. Blazevic, S. Giordano, and J.-Y. Le Boudec. Self Organized Routing in Wide Area Mobile Ad-Hoc Network. In *IEEE Symposium on Ad-Hoc Wireless Networks (Globecom 2001)*, November 2001.
2. L. Blazevic, S. Giordano, and J.-Y. Le Boudec. Self Organized Terminode Routing. *Cluster Computing Journal*, April 2002.
3. P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless newtorks. *3rd Int. Workshop on Descrete Algorithms and methods for mobile computing communications (DIAL M)*, August 1999.
4. F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers and Operational Research*, 13, 1986.
5. P. Hansen. The steepest ascent mildest descent heuristic for combinatorial programming. *Proc. Congr. on Numerical Method in Combinatorial Programming*, Academic Press, 1986.
6. Jon Kleinberg. The small-world phenomenon: an algorithmic perspective. *Technical Report 99-1776, Cornell Computer Science*, 1999.
7. Mobile Ad-hoc Networks (MANET) Working Group. http://www.ietf.org/html.charters/manet-charter.html.
8. M. Takai, L. Bajaj, R. Ahuja, R. Bagrodia, and M. Gerla. GloMoSim:A Scalable Network Simulation Environment. *Technical Report 990027, UCLA, Computer Science Department*, 1999.
9. D. J. Watts. In *Small Worlds, The dynamics of networks between order and randomness*. Princeton University Press, 1999.