DDR-Distributed Dynamic Routing Algorithm for Mobile Ad hoc Networks

Navid Nikaein, Houda Labiod and Christian Bonnet
Institut Eurécom *
2229, Route des Crêtes
B.P. 193
06904 Sophia Antipolis, France
{Navid.Nikaein, Houda.Labiod, Christian.Bonnet}@eurecom.fr

Abstract— **This paper presents an alternative simple loop-free bandwidth-efficient distributed routing algorithm for mobile ad hoc networks, denoted as distributed dynamic routing (DDR). Although DDR benefits from classical concepts like zone and forest, unlike previous solutions it achieves several goals at the same time. Firstly, it provides different mechanisms to drastically reduce routing complexity and improve delay performance. Secondly, it is an infrastructureless in a strong sense: it does not even require a physical location information. Finally, zone naming is performed dynamically and broadcasting is reduced noticeably.**

Keywords—**Ad hoc networks, graph-theoretic terminology, routing.**

I. INTRODUCTION

The growth of wireless communications coupled with high-speed broadband technology has led to a new era in telecommunications. Actually, in third generation mobile networks, efforts are undertaken to merge many technologies and systems to support a wide range of traffic types with various quality-of-service requirements. As wireless communication channels are highly affected by unpredictable temporal/spatial factors like co-channel interference, adjacent channel interference, propagation path loss and multipath fading, it is necessary to embed various adaptive mechanisms making these systems self-adaptive and more efficient, satisfying application best requirements and mitigating bad effects of wireless channels. However, there are some applications where the fixed infrastructure may not be present or is too expensive to maintain. So in this case, it is better to apply infrastructureless architecture for the desired system. Globally, future mobile networks can be classified in two main types: infrastructure and infrastructureless mobile networks, known as mobile ad-hoc networks. Many critical issues have to be addressed in the both systems. This paper focuses on the routing issue of mobile ad-hoc networks.

A Mobile ad hoc network (Manet) [1] is a set of wireless mobile hosts/nodes (MNs) forming dynamic autonomous networks. MNs communicate with each other without the intervention of a centralized access point or base station. No infrastructure is required. Routes between two nodes consist of hops through other nodes in the network. Therefore, each MN takes part in discovery and maintenance of routes to other nodes. The main characteristics of Manet strictly depend on both wireless link nature and node mobility features. Basically, they include dynamic topology, bandwidth and energy constraints, security limitations, self-operated (stand-alone) and lack of infrastructure [2]. Manets are viewed as suitable systems which can support some specific applications [3]:

- Virtual classrooms,
- Military communications,
- Emergency search and rescue-operation,
- Data acquisition in hostile environments,
- Communication set-up in exhibitions, conferences, presentations, meetings, lectures, etc.

Because mobile ad hoc networks constitute a distributed multi-hop network characterized by a time-varying topology, limited bandwidth and limited power, conventional routing protocols are not appropriate to use. Therefore, it is necessary to develop new protocols able to ensure a correct reception of transmitted information on radio links and to determine efficiently routes to reach the desired destinations. In fact, to reach the challenge of responding to time and space variations of mobile environments, efficient powerful adaptive routing protocols must be designed with an aim of enhancing the overall system performance. The expected routing protocols should have some key features summarized as below [4]:

- Minimum hop and delay to destination,
- Fast adaptability to link changes,
- Stable routes selection,
- Distributed operation,
- Loop avoidance.

Designing a new routing algorithm may necessitate examining the main strengths of each tendency and comparing the different existing approaches [3][5][6]. In the literature related to routing schemes used in mobile ad hoc systems, we find a classification of various design choices (see Fig. 1):

1. Proactive versus reactive [7] versus hybrid approach,
2. Flat versus hierarchical architecture [7],
3. Global Position (GP) versus Global Position-Less (GPL) based protocols.

In proactive or table-driven routing protocols, each node continuously maintains up-to-date routing information to reach

* Eurécom's research is partially supported by its industrial partners: Ascom, Swisscom, Thomson-CSF, IBM France, CEGETEL, Motorola, France Telecom, Hitachi Europe and Texas Instruments.

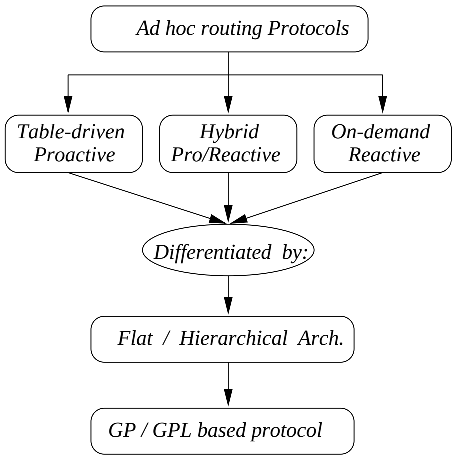

Fig. 1. Classification of ad hoc routing protocols

every other node in the network. Routing table updates are periodically transmitted throughout the network in order to maintain table consistency. Thus, the route is quickly established without delay. However, for highly dynamic network topology, the proactive schemes require a significant amount of resources to keep routing information up-to-date and reliable. In contrast to proactive approach, in reactive or on-demand protocols, a node initiates a route discovery throughout the network, only when it wants to send packets to its destination. In reactive schemes, nodes maintain the routes to active destinations. A route search is needed for every new destination. Therefore, the communication overhead is reduced at expense of delay due to route research. Furthermore, the rapidly changing topology may break an active route and cause subsequent route searches [7]. Finally, in hybrid protocols each node maintains both the topology information within its zone (coverage area), and the information regarding neighboring zones.

In hierarchical architectures, aggregating nodes into clusters and clusters into super-clusters conceals the details of the network topology. Some nodes, such as cluster heads and gateway nodes, have a higher computation communication load than other nodes. Hence, the mobility management becomes complex. The network reliability may also be affected due to single points of failure associated with the defined critical nodes. However, control messages may only have to be propagated within a cluster. Thus, the multilevel hierarchy reduces the storage requirement and the communication overhead of large wireless networks by providing a mechanism for localizing each node. On the contrary, in flat architectures, all nodes carry the same responsibility. Flat architectures do not optimize bandwidth resource utilization in large networks because control messages have to be transmitted globally throughout the network. The scalability gets then worse when the number of nodes increases significantly. However, the flat architectures are more desirable than hierarchical architectures, since they are more flexible and simple [7] [8].

In global position (GP) based protocols, the network relies on another system which can provide the physical information of the current position of MNs. Such physical locations can be obtained using the Global Position System (GPS) [9]. This involves an increase in the cost of the network maintenance. Generally, satellites are used to deliver this physical information. Any problem in one of the used satellites will surely affect the efficiency of the network and in some cases can easily make this latter blocked. In global position-less (GPL) based protocols, the network is stand-alone in the sense that it operates independently of any infrastructure. However, there are some situations where the physical location remains useful such as emergency disaster relief after a hurricane or earthquake.

The paper is organized as follows. Section II gives a brief overview on mobile ad hoc routing protocols. A comparison with ZRP, ZHLS and DST protocols is given. Section III describes in detail the six phases of the algorithm. The proof of a forest construction for any network is also outlined. Finally, Section IV provides concluding remarks and highlights some future work.

II. Related Work

Node mobility can cause frequent unpredictable topology changes. Hence finding and maintaining routes in Manets is non trivial task. Many protocols have been proposed for mobile ad hoc networks with the goal of achieving efficient routing. Several table-driven protocols have been proposed such as dynamic-destination-sequenced distance-vector (DSDV) [10], wireless routing protocol (WRP) [11], global state routing (GSR) [12], clusterhead gateway switch routing (CGSR) [13], fisheye state routing (FSR) and hierarchical state routing (HSR) [14]. Routing performed in DSDV and WRP is based on flat architecture while in HSR, FSH and CGSR, it is based on hierarchical architecture. Among the developed on-demand protocols, we can find cluster based routing protocol (CBRP) [15], ad hoc on demand distance vector (AODV) [16], dynamic source routing (DSR) [17], temporally ordered routing algorithm (TORA) [18], associativity based routing (ABR) [19], signal stability routing (SSR) [20], location-aided routing (LAR) [21], and distributed spanning trees (DST) based routing protocol [22]. These latter protocols except CBRP, maintain flat architectures. The LAR protocol needs physical location information. Hybrid protocols combine proactive and reactive features, we can find zone routing protocol (ZRP) [23][24] and zone-based hierarchical LSR protocol (ZHLS) [7]. Like LAR, ZHLS requires physical location information. Both ZRP and ZHLS support 2-level hierarchical architecture.

We propose a *global position-less* hierarchical hybrid routing algorithm, denoted as distributed dynamic routing (DDR) protocol. The main idea is to construct a forest from a network topology in a distributed way by using only periodic message exchanges between nodes and their neighbors. Each tree of the constructed forest forms a zone. Then, the network is partitioned into a set of non-overlapping dynamic zones. Each node computes periodically its zone ID independently. Each zone is connected via the nodes that are not in the same tree but they are in the direct transmission range of each other. So, the whole network can be seen as a set of connected zones. Mobile nodes can either be in a router mode or non-router regarding its position in its tree. This allows a more efficient energy consumption strategy. Each node is assumed to maintain routing information *only to those nodes that are within its zone*, and information regarding *only its neighboring zones*.

Similar to ZRP and ZHLS, DDR is a hybrid approach based on the notion of zone. Unlike ZRP, in DDR, the zones are not overlapped. In ZRP, each node keeps up-to-date information like distance and route to all the nodes within its zone, while in DDR each node needs to know *only the next hop* to all the nodes within its zone. This, reduces routing information and bandwidth utilization. Different from ZHLS, DDR *does not require physical location information for routing*. In ZHLS, each node maintains the zone connectivity of the whole network, while in DDR, each node keeps *only the zone connectivity of its neighboring zones*. In DDR, the zone size increases and decreases *dynamically* which is not the case in ZHLS. Moreover in ZHLS, zone naming is carried out at the design phase, therefore each node can determine exactly at any time its zone ID by mapping its physical position to a predefined zone map. On the contrary, in DDR the zone name assignment is done dynamically by some selected zone members. DDR *avoids broadcasting* by sending only the necessary information embedded in beacons to the neighboring nodes. To sum up, DDR reduces maintenance cost and radio resource consumption overhead and leads to a stand-alone network. Finally, in DDR there is no concept of root (as in DST) which *prevents single points of failure*.

III. Distributed Dynamic Routing Algorithm

In this section, we present the basic idea of the DDR-algorithm. Then, we give the necessary preliminary definitions to describe the algorithm. The DDR-algorithm consists of six phases: preferred neighbor election, intra-tree clustering, inter-tree clustering, forest construction, zone naming and zone partitioning. Then, we prove that, for any graph (i.e. for any network topology) the algorithm constructs a forest in a distributed manner. Finally, we give the algorithm of DDR.

A. Basic Idea

The main idea of our proposed distributed dynamic routing (DDR) algorithm is to construct a forest from a network topology (i.e. graph G). Each tree of the constructed forest forms a zone [1]. Then, the network is partitioned into a set of non-overlapping dynamic zones, $z_1, z_2, ..., z_n$. Each zone z_i contains p mobile nodes, $n_1, n_2, ..., n_p$. Then, each node calculates its zone ID independently. Each zone is connected via the nodes that are not in the same tree but they are in the direct transmission range of each other. So, the whole network can be seen as a set of connected zones. Thus, each node n_u from zone z_i can communicate with another node n_v from zone z_j. The size of zones increases and decreases dynamically depending on some network features such as node density, rate of network connection/disconnection, node mobility and transmission power.

B. Preliminary Definitions

A Manet topology is represented by an arbitrary graph [2] $G = (V, E)$, where V is the set of mobile nodes, and E is the set of edges. An edge exists if and only if the distance between two MNs is less or equal than a fixed radius r. This r represents the radio transmission range which depends on wireless channel characteristics including transmission power. Accordingly, the neighborhood of a node x is defined by the set of nodes that are inside a circle [3] with center at x and radius r, and it is denoted by $N_r(x) = N_x = \{n_j | d(x, n_j) \leq r, x \neq n_j, \forall j \in N, j \leq |V|\}$, where x is an arbitrary node in graph G. Notice that node x does not belong to its neighborhood ($x \notin N_x$). The degree of a node x in G is the number of edges which are connected to x, and it is equal to $deg(x) = |N_r(x)|$. The graph $G = (V, E)$ is called a tree T if and only if G is connected and contains no cycles. A node is called a *dead end* node if it is a leaf node in the tree T (i.e. its degree equals to 1). A forest F is a graph whose connected components are trees. We assume that each mobile node x generates periodically a message, known as the *beacon* B, to the neighboring nodes that are within its direct radio transmission range. The beacon is used to construct a forest in a distributed way. The time intervals between two beacons should depend on some network features like node mobility and rate of network connections/disconnections. There are five fields in a beacon: Zone ID number (ZID), Node ID number (NID), degree of NID (NID_DEG), my preferred neighbor (MY_PN) and preferred neighbor(s) (PN). The beacon B of node x is shown in Fig. 2, and it is denoted by B_x.

ZID	NID	NID_DEG	MY_PN	PN

Fig. 2. Fields of a beacon

The ZID identifies each tree from other trees. Each node initiates its ZID to its NID. A ZID is determined depending on ID numbers of some selected nodes belonging to the same

[1] We will use the terms tree and zone interchangeably.
[2] Here, the term graph means an undirected graph.
[3] We assume that MNs are moving in a two-dimensional plane.

tree as x. The way in which these nodes are selected, will be explained in section III-C.5. The ZID is also used to distinguish the nodes that are not in the same tree but they are in the direct transmission range of each other. These nodes are called *gateway* nodes, and the edge that connects two gateway nodes is called *bridge*. A node is in a *non-router* mode, if it is a non-gateway dead end node. Each MN is identified by a unique ID number, called NID. This NID can be the IP address. The NID_DEG is the associated degree of the NID in graph G. Each node calculates its degree just by adding the number of different received beacons, that is $deg(x) = |B_{N_r(x)}|$ [4]. The MY_PN flag distinguishes two different modes: PN election mode and PN(s) forward mode. The PN election mode indicates whether the preferred neighbor is determined by x; in this case this flag is set to 1. The PN(s) forward mode indicates that node x notifies the nodes belonging to its tree about *new* or *removed* member(s); in this case the flag is set to 0. The preferred neighbor of x is the node that owns preferred characteristics among neighboring nodes of x. Each node elects only one PN and can be chosen as the PN of many nodes. The way in which a PN is chosen will be explained in section III-C.1. Each node in the network maintains two tables: intra-zone table and inter-zone table. Intra-zone table keeps the information regarding the nodes of a tree. It contains two fields: node ID number (NID) and learned preferred neighbors (Learned_PN). The intra-zone table of node x is shown in Fig. 3, and it is denoted by $Intra\_ZT_x$. The $Intra\_ZT_x$ gives the current view of node x concerning its tree, and it is updated upon receiving beacons.

NID	$Learned\_PN$

Fig. 3. Intra-zone table

The field NID represents the ID number of each node that holds a tree edge with node x directly. So, the number of entry in intra-zone table gives the current degree of a node in the tree. The field Learned_PN represents the nodes that are reachable indirectly by their associated NID in the intra-zone table. On the contrary, inter-zone table keeps the information concerning neighboring zones of the zone in which node x belongs to. The inter-zone table of node x is shown in Fig. 4, and it is denoted by $Inter\_ZT_x$.

$GNID$	$NZID$	$Z\_Stability$

Fig. 4. Inter-zone table

Each entry in $Inter\_ZT_x$ contains the ID number of a gateway node (GNID), the zone ID of this gateway node, i.e. neighboring ZID (NZID) and the stability of this neighboring zone regarding node x (Z_Stability).

[4]Theoretically, this value has to be equal to $|N_r(x)|$.

C. Distributed Dynamic Routing Algorithm Description

C.1 Preferred Neighbor Election

Let x and y be any nodes of the graph $G = (V, E)$. We assume that initialy each node x knows the ID numbers and the degree of its neighboring nodes.[5] Based on these two information, node x can determine its PN. The node x searches a set of nodes whose degrees are equal to maximum neighborhood degree. This set is denoted by $PN_x = \{y|deg(y) = \max(deg(N_x))\}$. We distinguish three cases.

1. If the set is empty, then node x has no PN which means it has no neighbors. In Fig. 5, node n has no neighbor and consequently no PN.
2. If the PN_x has only one member, then this member is the elected PN. For example, in Fig. 5, node k has four neighbors: f, c, d, y, but the set of PN_k has only one member which is the node f.
3. The set of PN_x can have more than one member which is the case for node d, since $PN_d = \{k, c\}$. This means that there are more than one neighbor with the maximum neighborhood degree. In this case, we assume that node x elects a node with the greatest ID number. So, node d elects node k since its ID number is greater than node c (regarding to the alphabetical order).

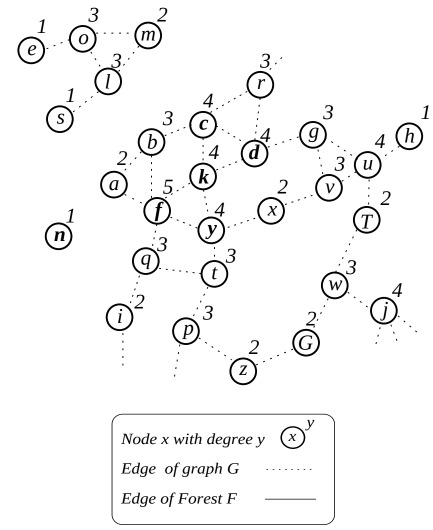

Fig. 5. An arbitrary graph G, where each node is characterized by its degree and a letter which represents its ID number. Assume that each node knows the ID numbers and the degrees of its neighboring nodes.

Each node x always wants to create an edge between itself and one of the nodes whose degree is equal to maximum neighborhood degree. Thus, the way in which a node is elected follows a monotonic increasing function depending on its degree and on its ID number. A forest is built after connecting each

[5]These informations are periodically provided in a beacon.

node to its PN. In section III-D (see theorem III-D) we will prove that, whatever is the network topology, this approach always yields a forest (i.e. we have no cycle).

C.2 Intra-Tree Clustering

As soon as node x determines its PN y, it must notify its neighboring nodes, especially y, of its decision. Therefore, node x sets its beacon to $B_x = (ZID, x, deg(x), 1, y)$. Then, node x updates its intra-zone table regarding y. Upon receiving x's beacon, each node updates its information regarding x and verifies whether they have been chosen as the PN of x. Among the neighboring nodes of x, the PN y forwards x's decision to nodes that hold a tree edge with y [6] by setting its beacon to $B_y = (ZID, y, deg(y), 0, x)$. If node y is chosen as the PN of many nodes at the same time, then y forwards their decisions encapsulated in PN field in a beacon, that is $B_y = (ZID, y, deg(y), 0, x : x' : x'' : ...)$. Notice that, the MY_PN flag distinguishes two different modes: PN election mode and PN(s) forward mode. Other neighboring nodes of x, add y to their own intra-zone table if node x already exists in their intra-zone tables. In this way, we say that y is learned to be the PN of x. Note that node x is also learned by the neighboring nodes of y. Node x does not need to recalculate its PN unless the $Intra\_ZT_x$ changes. If the PN of x remains unchanged, the PN field of x's beacon will only contain the set of PN learned by x, this set is denoted by $Learned\_PN_x$. Node x forwards the $Learned\_PN_x$, if it is not a dead end node. Consequently, each node does not add a node to its intra-zone table without knowing or learning. In this way, all the nodes belonging to the same tree are informed about the existence of other nodes. Another possible scenario occurs when the node y can not afford to be the x's preferred neighbor since it does not have enough energy or it has already accepted to be a PN of many nodes. Therefore, node y can decrement its degree regarding its neighborhood, so that it will be chosen by less neighboring nodes as a PN.

For example in Fig. 6, node k elects node f as its PN. Then node k generates $B_k = (ZID, k, 4, 1, f)$, and updates its intra-zone table. Notice that node k sets MY_PN field to 1. So, node f can be learned by nodes c, d. Upon receiving k's beacon, node f updates the information regarding node k in $Intra\_ZT_f$. Assume that the PN of f remains unchanged, that is node y. Since nodes b, a, q, y, k choose node f as their PN (highest degree priority), the node f must forward their decisions encapsulated in PN-field by setting its beacon to $B_f = (ZID, f, 5, 0, b : a : q : y : k)$. Therefore, nodes b, a, q, y are learned by node k, i.e. $Learned\_PN_k = b, a, q, y$. Consequently, if the PN of node k remains unchanged, the PN field of k's beacon will only contain $Learned\_PN_k$. So, the set of $Learned\_PN_k$ will be learned by the node c, d as well. But,

[6]These nodes dwell in the first column of intra-zone table of node y, i.e. $Intra\_ZT_y.NID$.

node c, d do not forward their learned PN, since they are dead end nodes. Also, nodes a and b are in the non-router mode, because they are non-gateway dead end nodes. Fig. 6 shows the constructed forest. Table I illustrates intra-zone tables of nodes f and k, when their tree is established.

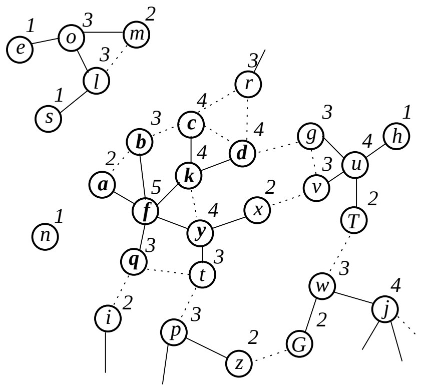

Fig. 6. The constructed forest

TABLE I
INTRA-ZONE TABLE OF NODES k AND f REGARDING FIG. 6

NID	$learned\_PN$
f	a, b, q, y, t, x
c	-
d	-

(a) $Intra\_ZT_k$

NID	$learned\_PN$
y	x, t
k	c, d
b, a, q	-

(b) $Intra\_ZT_f$

As shown in table I, the view of node x about its tree consists of two levels: NID, $Learned\_PN$. The NID level contains the nodes holding tree-edges with node x, i.e. node x can reach them directly. The second level $Learned\_PN$ contains the nodes that are learned by the NID level. In fact, node x can reach them via their associated NID in its intra-zone table. Therefore, node x only knows the next hop for its second level nodes. Actually, each entry in $Intra\_ZT_x$ can be seen as a branch of x, that is $NID \rightarrow Learned\_PN$. Thus, each node obtains a partial view of its tree in the sense that it does not know its detailed tree structure. For example in Fig. 6, consider the scenario where node k wants to communicate to one of the nodes belonging to its tree. According to its intra-zone table (see table I(a)), node k can reach the nodes a, b, q, y, t, x through node f, while other nodes f, c, d are directly reachable. So, regarding to $Intra\_ZT_k$, the next hop to reach the nodes a, b, q, y, t, x is the node f and not c, d. Although the nodes c, d can receive k's packets, they can simply drop them, since c and d are not defined as the next hop. Fig. 7 shows the view of node k on its tree.

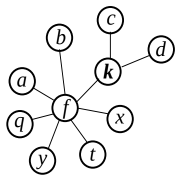

Fig. 7. View of node k on its tree

Once node x determines its *new* PN $y\prime$,[7] it must both update its intra-zone table and notify the remained members of its tree about the removed members. For this purpose, node x cuts the whole branch corresponding to its old PN y in its intra-zone table, that is $y \rightarrow Learned\_PN_y$. Then, node x forwards the set of removed members only to the remained nodes in the NID field of its intra-zone table by setting its beacon to $B_x = (-1, x, deg(x), 0, y \rightarrow Learned\_PN_y)$. Then, the remained nodes also forward the removed members if they are not dead end nodes. The $ZID = -1$ means that each node has to remove the specified nodes in PN-field of the beacon, and recalculates its ZID regardless to the removed members. The x's old PN y must carry out the same processing as x. For example in Fig. 6, if the link between nodes k and f is broken, then node k removes node f and its learned PNs $f \rightarrow a : b : q : y : t : x$ from its intra-zone table (see table I(a)). Then, node k forwards the beacon $B_k = (-1, k, deg(k), 0, f \rightarrow a : b : q : y : t : x)$ to the remained nodes c and d in the NID field of its intra-zone table. The nodes c and d do not forward the beacon, since they are dead end nodes. Also, node f cuts the branch corresponding to $k \rightarrow c : d$, and notifies the b, a, q, y about cut branch, and so on.

C.3 Inter-Tree Clustering

Each node x encounters two cases during the construction of its tree. Firstly, it can succeed to add some nodes to its tree and updates its intra-zone table. Otherwise, node x puts the remaining nodes in its inter-zone table. These nodes are considered as gateway nodes and they will be moved from the inter-zone table to intra-zone table whenever they can join x's tree. Node x increments the Z_Stability of a gateway node g_1 in its inter-zone table if the currently received ZID of g_1 is similar to its already existed ZID in $Inter\_ZT_x$. The stability of a zone depends directly on the ZID number. Section III-C.5 provides a detailed explanation of how a node determines its ZID and when it increments the Z_Stability of its neighboring zone. For example, in Fig. 9, node c belongs to the inter-zone table of node d until node d is informed about the existence of c in the tree. In fact, this information is provided by node k. After that, node d moves node c from $Inter\_ZT_d$ to $Intra\_ZT_d$. Table II shows the inter-zone table of node d, when the tree is established (see Fig. 9).

[7]This occurs when either the x's old PN y is no more available or there is a new node in the neighborhood of x whose degree (or ID number) is greater than y.

TABLE II
INTER-ZONE TABLE OF NODE d

$GNID$	$NZID$	$Z\_Stability$
r	z_4	++
g	z_5	++

C.4 Forest Construction

A forest is built by connecting each node to its PN. In section III-D (see theorem III-D) we will prove that, whatever is the network topology, this approach always yields a forest (i.e. we have no cycle). There is another possibility to construct a forest using the minimum neighborhood degree instead of maximum neighborhood degree. Although, it guarantees the construction of a forest, this strategy has not been adopted since the forest is constructed in the area where there is less node connectivity. In fact, nodes with a high degree of connectivity can not intersect themselves in the forest. For example in Fig. 8, the nodes $f\&k, y, c, d$ belong to different trees.

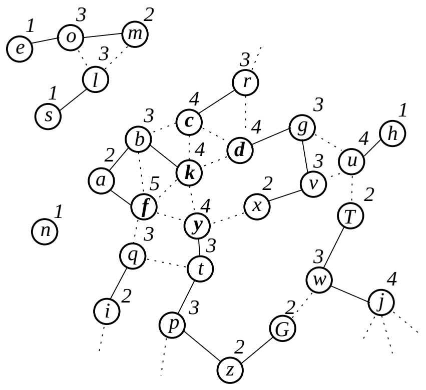

Fig. 8. Constructed forest with minimum neighborhood degree

C.5 Zone Naming

The objective here is to demonstrate that each node computes its zone name independently, and all the zone members arrive to the nearly same results. The zone name should depend on some zone characteristics such as ID number, node degree or stability of a node [19] during its life time in the zone. The choice of ID number is much simpler and does not require to maintain other information in intra-zone table. However, node degree and node stability are more pertinent than ID number regarding zone characteristics. Moreover, the zone name should not vary so often regarding rate of connection/disconnection of nodes. For this purpose, node x selects a subset s of the set of nodes in its intra-zone table for assigning a name or more precisely an ID number to the zone. Therefore, node x selects q highest ID numbers in its intra-zone table. One way to estimate the variable q can be $q = \lfloor \frac{n}{d} \rfloor$, where n is the number of bits in ID number and d is the compression degree of the hash function[8] . If the cardinality of a zone z_i is less than q then the selected nodes re-use their ID numbers respectively. Then, node x computes a hash function on the ID number of each selected node separately. Then, node x concatenates all the hashed ID numbers. The outcome of this concatenation gives the ZID. For example, in Fig. 9, if we assume that the $n = 12$ and $d = 3$ then $q = 4$, so node k selects the four nodes y, x, t, q. Notice that, node x selects these nodes in an ascending way. Node k determines its ZID by computing $h(y)|h(x)|h(t)|h(q)$, where $|$ denotes concatenation. In this way, we obtain the same number of bits in ZID as in NID. Clearly, each zone z_i will change its ID number over time. If we denote ZID_{t_1} the ZID at time t_1 and the ZID_{t_0} the ZID at time t_0 where $t_0 < t_1$, then the node uses a similarity function based on Euclidean distance to update the value of $Z\_Stability$. This function is called $Dsimilarity$ where $Dsimilarity(ZID_{t_1}, ZID_{t_0}) \leq d_{critic}$. If the distance between two ZIDs is far from the critical Euclidean distance, the node sets the value of $Z\_Stability$ to 1, otherwise, this value is incremented. In the both cases the node updates the ZID. So, we can conclude that the zone stability is strictly related to ZID number. Indeed, ZID determination is based on some randomly chosen NIDs in a tree. It therefore identifies the zone and it can simply reflect the zone stability. Fig. 9 shows the situation where each node assigns a name to its tree.

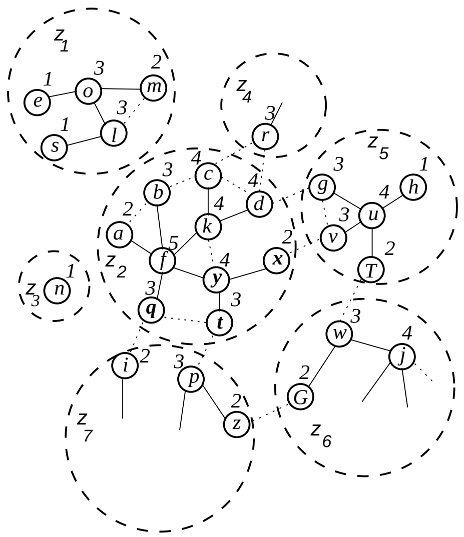

Fig. 9. Zone partitioning

[8] A hash function projects a value from a set s_1 with many (or even an infinite number of) members to a value from a set s_2 with a fixed number of (fewer) members. So, the compression degree of a hash function is the ratio of $|s_1|$ to $|s_2|$, that is $d = \lfloor \frac{|s_1|}{|s_2|} \rfloor$.

C.6 Zone Partitioning

The forest associated to the network contains a set of trees, $T_1, T_2, ..., T_k$. Each tree forms a zone. Then, the network is partitioned into a set of non-overlapping dynamic zones, $z_1, z_2, ..., z_n$. The zones are connected via gateway nodes. So, the whole network can be seen as a set of connected zones. The size of zone increases and decreases dynamically depending on some network features such as node density/mobility, rate of network connection/disconnection and transmission power.

D. Theorem

For any graph G (i.e. networks topology), let G' be the subgraph obtained by connecting each node to its PN. Then G' is a forest.

Proof: Let $G = (V, E)$ be the original graph and let $G' = (V, E')$ be the graph obtained by running a copy of the algorithm in Fig. 6 for each node $x \in V$. We first recall that the main idea of the algorithm is to select, for each node x in G, a neighbor that has maximum degree. Moreover, if more than one neighbor has maximum degree (i.e. $|PN_x| > 1$) then we select the one with maximum ID number. In order to prove that G' does not contain any cycle, we consider the function:

$$label(x) = \max_{w \in PN(x)} \{deg(w)|NID\}$$

where $|$ represents concatenation. We prove that G' cannot contain any cycle $C = x_1, \ldots, x_k, x_1$. Suppose the contrary, and let x_i be the vertex of C with the smallest value of $label(\cdot)$. Notice that such a vertex is unique.

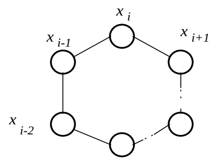

Fig. 10. The proof of theorem III-D

Let us consider the two vertices of x_{i-1} and x_{i+1} adjacent to x_i in C (see Fig. 10). Without loss of generality, assume that the algorithm in Fig. 10 chooses an adjacent vertex x_{i+1} (if neither x_{i-1} nor x_{i+1} are chosen, then C is not a cycle). Consider now the execution of the algorithm on x_{i-1}. We show that such a node will not choose x_i, thus implying that C is not a cycle. Indeed, by the choice of x_i, the vertex x_{i-2} adjacent to x_{i-1} satisfies one of the following two conditions:

1. $deg(x_{i-2}) > deg(x_i)$. In this case $x_i \notin PN_{x_{i-1}}$
2. $deg(x_{i-2}) = deg(x_i)$ and $NID(x_{i-2}) > NID(x_i)$. If $x_i \in PN(x_{i-1})$, then also $x_{i-2} \in PN_{x_{i-1}}$ (otherwise x_i is not selected as neighbor of x_{i-1}). Since $label(x_i) < label(x_{i-2})$, it holds $NID(x_i) < NID(x_{i-2})$. So, vertex x_{i-1} will select x_{i-2}.

This proves the theorem. ■

E. DDR-Algorithm

The outline of the algorithm used to construct a distributed forest is formally stated below. The details of the called functions can be found in appendix. This algorithm has to be executed at each node in the network.

```
The DDR-Algorithm executed in node $x$
Let $G = (V, E)$ be a graph.
Let $x$ be any node of $G$.
Let $B_x$ be the beacon of node $x$.
Let $N_x$ represents the set of neighboring nodes of $x$.
Let $B_{N_x}$ represents a set of beacons from $N_x$.
Let $PN_x$ be the set of nodes with $\max(deg(N_x))$.
Let $EPN_x$ be the elected preferred node of $x$.
Let $EPN_x.old$ be the old elected preferred node of $x$.
Let $Intra\_ZT_x$ represents the intra-zone table of node $x$.
Let $Inter\_ZT_x$ represents the inter-zone table of node $x$.
Let $ZID$ be the zone ID number of node $x$, initialized to $x$.

$Begin$
  $Buffer = ReceiveB_{N_x}()$;
  $deg(x) = |B_{N_x}|$;
  $EPN_x = Determine\_PN(PN_x)$;
  Send $B_x = (ZID, x, deg(x), 1, EPN_x)$ ;          $(1)$
  $Update\_IntraZT(EPN_x)$;
  $Learned\_PN_x[] = Build\_IZT(Buffer, EPN_x)$;
  $ZID = Zone\_Name(Intra\_ZT_x)$;
  $if(Learned\_PN_x[] \neq \emptyset \,\&\&\, |Intra\_ZT_x.NID| > 1)(2)$
      Send $B_x = (ZID, x, deg(x), 0, Learned\_PN_x[])$
  $EPN_x.old = EPN_x$;
  $Clear(Buffer)$;
$End.$
$(1)$ PN election mode.
$(2)$ Forward mode.
```

Fig. 11. DDR-Algorithm

IV. Conclusion

This paper presents an alternative routing algorithm for mobile ad hoc networks, called DDR. It is based on forest construction of mobile nodes. To design a new routing algorithm for Manet, it is necessary to take into account some important factors like bandwidth/energy constraints and node mobility. To reach this goal, DDR combines two main notions: zone and forest. Zones are used in order to reduce the delay due to routing process and to reach high scalability. Forest gives an appropriate structure to the mobile ad hoc network that allows a better radio resource utilization. The non-overlapped zones are dynamically constructed in relation with the forest. For future work, we will address the routing protocol description. A performance analysis will be carried out to make a comparison between DDR and some other protocols.

Acknowledgment

The authors wish to thank Paolo Penna, Neda Nikaein and Sergio Loureiro for their useful discussions.

References

[1] *Mobile Ad Hoc Networking (MANET)*, Available at $http://tonnant.itd.nrl.navy.mil/manet/manet\_home.html$.

[2] E. R. Yedavalli, "Implementation of wireless multipath routing protocol (WMRP) for RDRN," Unpublished, available at $http://www.ittc.ukans.edu/\ eshwar/eecs845/index.html$.

[3] E. M. Rover and C. K. Toh, "A review of current routing protocols for ad hoc mobile wireless networks," *IEEE Personal Communications*, vol. 6, no. 2, pp. 46–55, 1999.

[4] C-K. Toh, "Associativity-based routing for ad-hoc mobile networks," *Wireless Personal Communications Journal*, vol. 4, no. 2, 1997.

[5] S-J. Lee, M. Gerla, and C-K. Toh, "A simulation study of table-driven and on-demand routing protocols for mobile ad hoc networks," *IEEE Network*, vol. 13, no. 4, pp. 48–54, 1999.

[6] J. Broch, D. A. Maltz, D. B. Johnsin, Y-C. Hu, and J. Jetcheva, "A performance comparison of multi-hop wirless ad hoc network routing protocols," in *Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking*, MOBICOM'98.

[7] M. Joa-Ng and I-Tai Lu, "A peer-to-peer zone-based two-level link state routing for mobile ad hoc networks," *IEEE on Selected Areas in Communications*, vol. 17, no. 8, pp. 1415–1425, 1999.

[8] P. Sass D. Baker, M. S. Corson and S. Ramanathan, "Flat vs. hierarchical network control architectures," *ARO/DARPA Workshop on Mobile Ad-Hoc Networking*, 1997.

[9] B. Parkinson and S. Gibert, "NAVSTAR: Global positioning system - ten years later," in *Proceeding of IEEE*, 1983, pp. 1177–1186.

[10] C. E. Perkins and P. Bhagwat, "Highly dynamic destination sequenced distance-vector routing (DSDV) for mobile computers," in *SIGCOMM'94*.

[11] S. Murthy and J.J. Garcia-Luna-Aceves, "An efficient routing protocol for wireless networks," *ACM Mobile Networks and Applications Journal*, 1996.

[12] T. Chen and M. Gerla, "Global state routing: A new routing scheme for ad-hoc wireless networks," in *Proc. IEEE ICC'98*.

[13] C-C. Chiang, "Routing in clustered multihop, mobile wireless networks with fading channel," in *IEEE SICON'97*, pp. 197–211.

[14] A. Iwata, C-C. Chiang, G. Peiand M. Gerla, and T.-W. Chen, "Scalable routing strategies for ad hoc wireless networks," *IEEE Journal on Selected Areas in Communications, Special Issue on Ad-Hoc Networks*, vol. 17, no. 8, pp. 1369–1379, 1999.

[15] M. Jiang, J. Li, and Y. C. Tay, "Cluster based routing protocol," IETF Draft, 1999.

[16] C. E. Perkins, E. M. Royer, and S. R. Das, "Ad hoc on-demand distance vector routing," IETF Draft, 1999.

[17] D. B. Johnson and D. A. Maltz, *Dynamic Source Routing in Ad Hoc Wireless Network*, Kluwer Academic Publishers, 1996.

[18] V. D. Park and M. S. Corson, "A highly adaptive distributed routing algorithm for mobile wireless networks," in *INFOCOM'97*.

[19] C-K. Toh, "A novel distributed routing protocol to support ad-hoc mobile computing," *IEEE International Phoenix Conf. on Computers and Communications*, IPCCC'96.

[20] R. Dube, C. D. Rais, K. Wang, and S. K. Tripathi, "Signal stability based adaptive routing (SSR) for ad-hoc mobile networks," *IEEE Personal Communications*, vol. 4, no. 1, pp. 36–45, 1997.

[21] Y-B. Ko and N. H. Vaidya, "Location-aided routing in mobile ad hoc networks," *4th Annual International Conference on Mobile Computing and Networking*, MOBICOM'98.

[22] S. Radhakrishnan, N. S. V. Rao G. Racherla, C. N. Sekharan, and S. G. Batsell, "DST - a routing protocol for ad hoc networks using distributed spanning trees," *IEEE Wireless Communications and Networking Conference*, pp. 100–104, 1999.

[23] Z. J. Hass and M. R. Pearlman, "The zone routing protocol (ZRP) for ad hoc networks," Internet Draft, 2000.

[24] M. R. Pearlman and Z. J. Hass, "Determining the optimal configuration for zone routing protocol," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 8, pp. 1395–1414, 1999.

APPENDIX

1. Preferred Neighbor Election

```
Determine_PN($PN_x$){
        $if(|PN_x| = 0)$
            $return$ $"NoN"$; /*No Neighbor*/
        $elseif(|PN_x| = 1)$
            $return$ $PN_x.NID$;
        $else$
            $return$ $\max(PN_x.NID)$;
}
```

2. Intra and Inter Zone Table Construction

```
Build_IZT($B_{N_x}, EPN_x$){
  $for(i = 0; i < |B_{N_x}|; i++)\{$
        $if(B_x.ZID = -1)$
            $Remove\_IntraZT(B_{N_x}.PN)$;
        $if(B_{N_x}.PN = x$ && $B_{N_x}.MY\_PN = 1)\{$ /*Node $x$ is chosen as a PN*/
            $Learned\_PN_x[] = B_{N_x}.NID$; /*Add a node to the list of learned node*/
            $Update\_IntraZT(B_{N_x}.NID)$;
        }
        $elseif((B_{N_x}.NID \in Intra\_ZT_x$ || $B_{N_x}.PN \in Intra\_ZT_x)$ && $B_{N_x}.PN \neq x)\{$
            $if(B_{N_x}.PN \notin Intra\_ZT_x$ && $B_{N_x}.NID \in Intra\_ZT_x)\{$
                $if(B_{N_x}.MY\_PN = 1$ && $B_{N_x}.NID \neq EPN_x)\{$ /*Changing PN*/
                    /*Remove the entry corresponding to $NID$, and save it into temp*/
                    $temp = RemoveBranch\_IntraZT(B_{N_x}.NID)$;
                    /*Inform remained members of $Intra\_ZT_x$ about removed members saved in $temp$*/
                    $if(|Intra\_ZT_x.NID| > 1)$ /*$x$ is not a dead end node*/
                        Send $B_x = (-1, x, deg(x), 0, temp)$; /*Forward mode*/
                    $temp = 0$;
                }
                $else\{$ /*Learned PN*/
                    $Learned\_PN_x[] = B_{N_x}.PN$;/*Add a node to the list of learned nodes*/
                    $Update\_IntraZT(B_{N_x}.NID, B_{N_x}.PN)$;
                }
            }
            $elseif(B_{N_x}.NID \notin Intra\_ZT_x$ && $B_{N_x}.PN \in Intra\_ZT_x)\{$/*Learned NID*/
                $Learned\_PN_x[] = B_{N_x}.NID$; /*Add a node to the list of learned nodes*/
                $Update\_Intra\_ZT(B_{N_x}.PN, B_{N_x}.NID)$;
            }
        }
        $elseif(B_{N_x}.NID \notin Intra\_ZT_x$ && $B_{N_x}.PN \notin Intra\_ZT_x)\{$ /*Gateway Node Detection*/
            $if(B_{N_x}.NID \notin Inter\_ZT_x)$
                $Insert\_Inter\_ZT(B_{N_x}.NID, B_{N_x}.ZID, 1)$;
            $else\{$
                $if(Dsimilarity(Inter\_ZT_x[B_{N_x}.NID].NZID, B_{N_x}.ZID) \leq d_{critic})$
                    $Inter\_ZT_x[B_{N_x}.NID].Z\_Stability++$;
                $else$
                    $Inter\_ZT_x[B_{N_x}.NID].Z\_Stability = 1$;
                $Intra\_ZT_x[B_{N_x}.NID].NZID = B_{N_x}.ZID$;
            }
        }
  $return$ $Learned\_PN_x[]$;
  }
```

3. Zone Naming

```
Zone_Name($Intra\_ZT_x$){
  $for(i = 0; i < q; i++)$
          $sn[i] = \max(Intra\_ZT_x)$;
  return $h(sn[0])|h(sn[1])|...|h(sn[q-1])$;
}
```